**MASTER EN SCIENCES INFORMATIQUES**

# Transition de IPv4 vers IPv6

Mimbe, Benis Thierry

*Award date:*
2002

# Transition de IPv4 vers IPv6

MIMBE Benis Thierry

Mémoire présenté en vue
de l'obtention du diplôme
de licencié en informatique

Promoteur: Pr. O. Bonaventure

Année académique 2001-2002

*Nous tenons tout particulièrement à remercier notre promoteur, Monsieur Olivier Bonaventure pour sa relecture attentive. Sans ses précieux conseils, cette recherche n'aurait pu aboutir.*

*Nous exprimons également notre profonde gratitude à Monsieur Uhlig Steeve. Sa gentillesse et sa disponibilité se sont avérés de réels soutiens tout au long de ce travail.*

*Jamais nous ne remercierons assez ces personnes pour leurs commentaires et leurs remarques constructives.*

**Table des matières**

# Introduction

Bien que la technique CIDR défini dans le RFC 1519 permet de repousser l'échéance de le pénurie d'adresses IP de quelques années, tout le monde s'accorde à dire que les jours du protocole IP sont comptés. Hormis le problème technique, une autre interrogation apparaît. Jusqu'à récemment, le réseau Internet était utilisé largement par les universités, les industries de pointe et le gouvernement américain –surtout le Département de la défense–. Mais depuis quelques années, Internet intéresse de plus en plus les entreprises et les sociétés commerciales. De plus, il est certain que son expansion se poursuivra et qu'il sera utilisé par un plus grand nombre d'individus et de systèmes exprimant les uns les autres des besoins différents. Par exemple, des millions de personnes pourront utiliser des radio-ordinateurs portables pour rester en contact avec leur entreprise ou leur domicile. Avec la convergence imminente de l'ordinateur, chaque poste de télévision deviendra également un équipement d'accès à Internet permettant à des milliards d'individus de pratiquer, entre autres, la vidéo à la demande, le télé-achat et le commerce électronique. Dans ces circonstances, il est évident que le protocole IP devait évoluer et devenir plus flexible.

Conscients des problèmes à venir, l'IETF débuta en 1990 les travaux d'une nouvelle version du protocole IP qui, actuellement, sont toujours en cours. Cette version ne doit jamais être en rupture d'adresses, elle doit en outre résoudre une variété de problèmes nouveaux et offrir plus de flexibilité et d'efficacité. Les objectifs de ce nouveau protocole sont de:

- supporter des milliards d'ordinateurs, en se libérant de l'inefficacité de l'espace des adresses IP actuelles;
- réduire la taille des tables de routage;
- simplifier le protocole, pour permettre un routage rapide des datagrammes;
- fournir une meilleure sécurité –authentification et confidentialité– que l'actuel protocole IP;
- accorder plus d'attention au type de service, et notamment aux services associés au trafic temps réel;
- faciliter la diffusion multidestinataire en permettant d'en spécifier l'envergure;
- donner la possibilité à un ordinateur de se déplacer sans changer son adresse;
- accorder à l'ancien et au nouveau protocole une coexistence pacifique.

Le nouveau protocole qui est né de cette réflexion et qui répond aujourd'hui raisonnablement aux objectifs édictés est IPv6. La nécessité du déploiement de ce dernier s'avère réel.

L'objectif global de ce travail est précisément d'analyser les problèmes liés à la transition du réseau actuel basé sur IPv4 vers un réseau basé sur IPv6 et les solutions proposées afin de les résoudre. Pour traiter la problématique du passage de IPv4 à IPv6, nous choisissons de procéder en trois temps.

D'abord, nous décrivons succinctement dans un premier chapitre les spécifications du nouveau protocole IPv6 en les comparant à celles du protocole IPv4. Plus particulièrement, nous analysons les en-têtes de base des datagrammes et les extensions de ces deux protocoles. En ce qui concerne le protocole IPv6, nous étudions également la configuration automatique, processus qui lui est plus spécifique.

Ensuite, le second chapitre de notre travail traite des problèmes rencontrés au moment du passage de l'ancien au nouveau protocole et présente quelques techniques permettant une communication entre les deux différents types d'équipements. Pour ce faire, nous proposons une synthèse bibliographique de certaines solutions envisagées jusqu'à ce jour pour résoudre les problèmes de transition entre les protocoles IPv4 et IPv6.

Enfin, le troisième et dernier chapitre est consacré quant à lui à une analyse fonctionnelle et technique de trois solutions développées dans le second chapitre. Nous avons choisi d'étudier le protocole *NAT-PT*, **N**etwork **A**ddress **T**ranslation-**P**rotocol **T**ranslation, le protocole *NAPT-PT*, **N**etwork **A**ddress **P**ort **T**ranslation-**P**rotocol **T**ranslation et le protocole *DSTM*, **D**ual **S**tack **T**ransition **M**echanism.

# 1 Spécifications du protocole IPv6

Il nous semble opportun de commencer l'étude de la transition de IPv4 vers la nouvelle version par une comparaison du format des datagrammes de ce dernier avec ceux de l'ancien protocole. Un datagramme IP est constitué de deux parties principales: un *champ en-tête* et un *champ données*, ce dernier constituant la charge utile du datagramme.

L'en-tête de base des datagrammes IPv4 est représenté à la figure 1.1 et celle des datagrammes IPv6 à la figure 1.2.

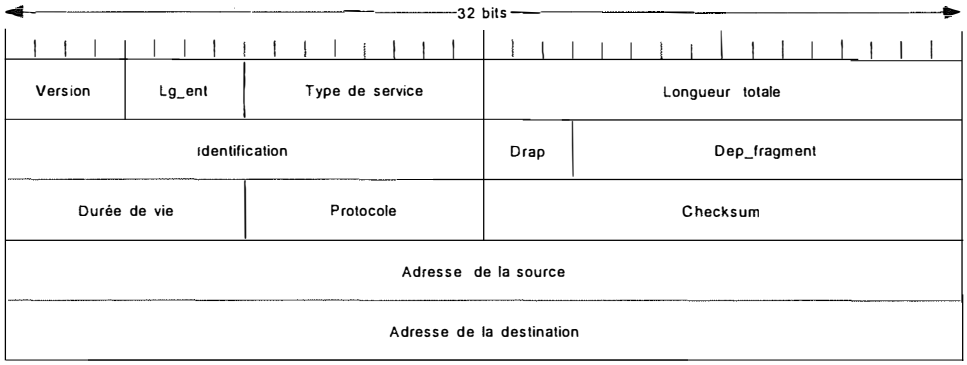



Fig. 1.1 - En-tête de base des datagrammes IPv4 [Tan97]

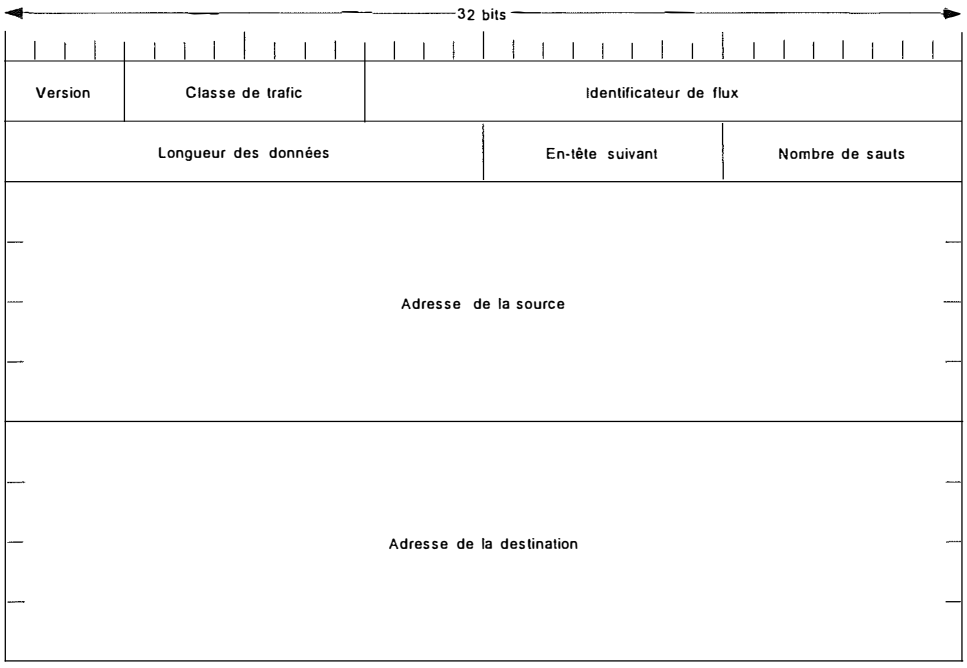



Fig. 1.2 - En-tête de base des datagrammes IPv6 [RFC2460]

## 1.1 Version

Le premier champ d'un datagramme IP *Version* contient sur 4 bits le numéro de version du protocole IP utilisé pour créer les datagrammes. En IPv6, ce champ est toujours égal à 6 –à 4 pour IPv4–. Il permet de vérifier que la source, le destinataire et tout routeur intermédiaire sont en accord sur le format du datagramme qu'ils manipulent. En indiquant le numéro de

version, il est possible d'utiliser des versions différentes du protocole IP dans un même réseau. Pendant la période de transition de IPv4 vers IPv6 qui durera probablement une décennie, les routeurs devront examiner ce champ pour savoir quel type de datagramme ils routent. Ce champ est le seul qui occupe la même place dans le paquet IPv6 et dans le paquet IPv4.

## 1.2 Classe de trafic

Le champ *Classe de trafic* codé sur 8 bits permet la différenciation de services conformément aux spécifications du RFC 2474. Il est l'équivalent du champ *ToS* –Type de service– dans le format d'en-tête des datagrammes IPv4 –figure1.1– où il est aussi appelé octet *DiffServ* –DS–. Ce champ permet une gestion différenciée des datagrammes en cas de congestion en indiquant au sous-réseau le type de service désiré. Il précise en outre comment les datagrammes doivent être gérés. C'est ainsi que différentes qualités de service et de mode d'acheminement peuvent être définis –par exemple, prépondérance du rythme de remise des datagrammes par rapport à la précision dans le cas de la voix numérisée ou transfert sans erreur lors de l'échange de fichiers–.

Le champ *Classe de trafic* est divisé en deux parties. Son format est repris à la figure 1.3. La deuxième partie du champ, composée de deux bits est inutilisée dans les deux versions du protocole IP. Mais dans la nouvelle version, il devrait servir aux routeurs pour indiquer un risque de congestion.

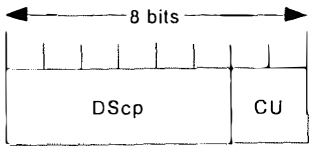



**Fig. 1.3** - Format de l'octet *classe de trafic*

Dans le cas d'IPv6, le premier sous champ du champ *Classe de trafic* baptisé *DScp* –DiffServ Code Point– contient les valeurs de différents comportements en cas de congestion du réseau. Sans différenciation, les paquets ont la même probabilité de rejet. Avec celle-ci, plusieurs classes de trafic seront définies. Les paquets appartenant aux classes les plus élevées auront une probabilité de rejet plus faible.

L'intérêt principal de la différenciation de service est qu'elle ne casse pas le modèle initial de l'Internet –version 4 ou version 6–. Il n'y a aucune garantie qu'un trafic d'une classe de service haute arrive à destination. Mais la probabilité est plus importante. L'autre intérêt des classes de service vient de la possibilité d'agrégation des flux. La classe d'appartenance est indiquée dans l'en-tête du paquet. Les applications peuvent marquer les paquets en fonction des paramètres locaux –trafic du PDG de la société, flux multimédia interactif…–. Le fournisseur d'accès qui récupère le trafic n'a plus à se préoccuper des applicatifs, il vérifie que le trafic d'une classe ne dépasse pas le contrat préalablement établi. Le fournisseur d'accès devra passer des accords avec les autres opérateurs pour pouvoir faire transiter les flux avec un traitement approprié.

En plus, l'octet *DS* a gardé pour des raisons de compatibilité avec des équipements existants la valeur du bit *ToS* qui était la plus fréquemment utilisée. En particulier, la valeur 0xE0 correspond à la classe de contrôle du réseau. Elle est utilisée dans la mise en œuvre d'IPv6 pour l'émission de certains paquets ICMPv6.

## 1.3 Identificateur de flux

C'est un champ qui permet à une source ou à une destination d'ouvrir une pseudo-connexion avec des exigences particulières. Par exemple, un flot de datagrammes provenant d'un ordinateur et destiné à un autre ordinateur pourrait avoir des exigences rigoureuses de délais d'acheminement et pourrait souhaiter disposer d'une certaine bande passante. C'est un champ qui contient un numéro unique choisi par la source. Ce numéro a pour but de faciliter le travail des routeurs et la mise en œuvre des fonctions de qualité de service. Cet indicateur peut être considéré comme un marqueur de contexte utile pour le routeur. Celui-ci peut alors faire un traitement particulier: choix d'une route, traitement en "temps réel" de l'information.

Avec IPv4, pour optimiser le traitement, certains routeurs se basent sur l'adresse de la source, sur l'adresse de la destination, sur les numéros de port de la source et de la destination, et sur le protocole pour construite un contexte. Ce contexte sert à router plus rapidement les paquets puisqu'il évite de consulter les tables de routage pour chaque paquet. Ce contexte est détruit après une période d'inactivité.

Avec IPv6, l'utilisation de ce champ n'est pas clairement définie à ce jour.

Il ne faut pas confondre le champ *Identificateur de flux* de IPv6 avec le champ *Identification* de l'en-tête des datagrammes IPv4. En fait dans IPv6, ce dernier concept est repris dans l'extension de fragmentation.

## 1.4 Longueur des données utiles -Payload-

En IPv4, le concept de longueur de l'information transmise est regroupé dans deux champs : le champ *Lg_ent* et le champ *Longueur totale.* L'en-tête d'un datagramme étant de longueur variable, le champ de 4 bits *Lg_ent*, indique sa longueur en mots de 32 bits. L'en-tête le plus courant ne contient pas d'options IP, il a une longueur de 20 octets. La valeur de *Lg_ent* est égale à 5 –5 mots de 32 bits–.

La valeur maximale de *Lg_ent* est de 15. Ceci impose une limite supérieure sur la taille de l'en-tête à 60 octets , ce qui correspond à 40 octets d'options. Pour certaines options, en l'occurrence celle qui enregistre la route empruntée par un datagramme, les 40 octets du champ sont insuffisants, ce qui rend le champ *Option* inutile.

Le champ *longueur totale*, codé sur 16 bits, indique la longueur en octets du datagramme IPv4 --en-tête + charge utile–. La taille maximale d'un datagramme IPv4 est, de ce fait, égale à $2^{16}$– 1, soit 65 535 octets. Pour la plupart des applications, cela ne constitue pas une contrainte sévère. Mais cela pourra le devenir, lorsque les réseaux à très haut débit pourront transporter des datagrammes géants de plus de 65 535 octets –cas des jumbogrammes–.

Contrairement à IPv4, le champ *longueur des données* de l'en-tête du datagramme IPv6 couvrant deux octets ne contient que la taille des données utiles sans prendre en compte la

longueur de l'en-tête. Pour des paquets dont la taille de données serait supérieure à 65 535, ce champ vaut 0 et l'option jumbogramme de l'extension proche-en-proche est utilisée.

## 1.5 En-tête suivant

Ce champ spécifie le type de l'en-tête suivant éventuel. La raison ayant conduit à une simplification de l'en-tête dans IPv6 réside dans une possible utilisation des en-têtes d'extension additionnels –optionnels–. Ce champ indique TCP ou UDP dans IPv4. Dans IPv6, c'est plus général: Il indique quel en-tête d'extension suit l'en-tête fixe parmi les 6 définis actuellement. Si l'en-tête est le dernier d'une suite d'en-tête IP, le champ *en-tête suivant* indique à quel protocole de transport le passer –par exemple ICMP, UDP, TCP…voir figure 1.4–.

Les extensions contiennent aussi ce champ pour permettre un chaînage.

| **Valeur** | **Protocole** |
|---|---|
| 6 | TCP |
| 17 | UDP |
| 41 | IPv6 |
| 58 | ICMPv6 |

**Fig. 1.4** - Valeur et signification du champ *en-tête suivant*

## 1.6 Nombre de sauts

Ce champ est utilisé pour empêcher les datagrammes de circuler indéfiniment dans le réseau Internet. A partir d'une valeur initiale, il est décrémenté à chaque nœud traversé. Il joue le même rôle que le champ *durée de vie* d'IPv4, à savoir représenter le nombre de sauts ou de pas –hops–. En théorie, dans IPv4, il y a une notion de temps en secondes mais aucun routeur ne l'utilisent comme tel. Le nom a changé pour refléter l'usage actuel. En pratique, les paquets ne restent que quelques millisecondes dans les routeurs et aisi la décrémentation est arrondie à 1. Dans IPv6, comme il s'agit de nombre de sauts, la décrémentation est toujours égale à 1. Un datagramme retransmis par un routeur est rejeté avec l'émission d'un message ICMPv6 vers la source, si la valeur après décrémentation atteint 0.

On peut noter une limitation puisque ce champ codé sur 8 bits n'autorise la traversée que de 255 routeurs. En réalité, dans l'Internet actuel, le nombre maximal de routeurs traversé est d'une quarantaine, ce qui laisse une bonne marge pour l'évolution du réseau.

## 1.7 Adressage en IPv6

Nous allons décrire maintenant la nouveauté majeure apportée par IPv6 à savoir l'adressage.

### 1.7.1 Structuration des adresses et agrégation

Un des problèmes majeur d'IPv4 est l'augmentation incontrôlée de la taille des tables de routage. Ce phénomène est dû à une mauvaise agrégation des adresses dans les tables. Il faudrait être capable de router des ensembles de réseaux identifiés par un seul descripteur. Les adresses IPv4 sont trop courtes pour permettre une bonne structuration.

Pour pouvoir assurer une bonne agrégation, il faut une structuration de l'Internet à différents niveaux. Le premier niveau est le site. Au niveau directement supérieur au site, on trouve les prestataires de connectivité IP. Au niveau le plus haut, on trouve de très gros prestataires qui s'occupent des transports intercontinentaux. Les prestataires de connectivité IP peuvent eux-mêmes être structurés en plusieurs niveaux, car un prestataire peut revendre la connectivité IP tout en étant client d'un autre prestataire.

Pour l'acheminement des datagrammes, le routeur doit avoir une vision complète de l'ensemble des niveaux inférieurs et fournir aussi une vision globalisée des niveaux supérieurs.

On peut donc distinguer différents niveaux conceptuels d'adressage qui sont:

- les grands transporteurs ;
- les fournisseurs de service ;
- le découpage fin des sites.

La taille des adresses IPv6 permet de définir des niveaux en attribuant un certain nombre de bits à chacun d'eux.

### 1.7.2 Durée de vie des adresses

Une adresse est attribuée pour une période de temps. La valeur par défaut est 60 heures. Cette période peut être prolongée ou portée à l'infini.

Une adresse peut se trouver dans l'un des trois états qui sont: préféré, déprécié et invalide.

On dit qu'une adresse est dans l'état "préféré", quand les applications peuvent l'utiliser pour leur communication. Le changement brusque d'une adresse peut entraîner une rupture brutale des communications TCP qui sont en train d'utiliser celle-ci comme identificateur de connexion. Un mécanisme d'obsolescence est mis en place pour invalider progressivement une adresse. Avant d'être désallouée, une adresse passe à l'état "déprécié". Les applications qui ont commencé à utiliser cette adresse lorsqu'elle était à l'état "préféré" peuvent continuer à le faire.

Lorsqu'une adresse passe à l'état "invalide", plus aucune application ne doit l'utiliser.

### 1.7.3 Notation

Une nouvelle notation a été définie pour décrire les adresses IPv6 de 16 octets. Elle comprend huit groupes de quatre chiffres hexadécimaux séparés par le caractère ":". Par exemple:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Puisque plusieurs adresses ont de nombreux zéros dans leur libellé, trois optimisations ont été définies. Tout d'abord, les premiers zéros d'un groupe peuvent être omis comme par exemple 0123 qui peut s'écrire 123. Ensuite, plusieurs champs nuls consécutifs peuvent être abrégés par un double deux points. C'est ainsi que l'adresse ci-dessus devient

8::0123:4567:89AB:CDEF

Pour éviter toute ambiguïté, l'abréviation"::" ne peut apparaître qu'une fois au plus dans une adresse.

Les adresse IPv4 peuvent être écrites en utilisant la représentation de l'adresse en notation décimale pointée précédée d'un double deux points comme par exemple:

::192.31.320.46

La notation des préfixes est similaire à la notation utilisée pour les préfixes IPv4. Un préfixe IPv6 est donc représenté par la notation

adresse-IPv6/longueur du préfixe-en-bits

### 1.7.4 Types d'adresses

Les adresses IPv6 sont classées en trois grands types: unicast, multicast et anycast.

Une adresse de type unicast désigne une interface unique. Un paquet envoyé à une telle adresse sera remis à l'interface ainsi identifié.

Une adresse de type multicast désigne un groupe d'interfaces qui appartiennent en général à des nœuds différents pouvant être situés n'importe où dans l'Internet. Lorsqu'un paquet a pour destination une adresse de type multicast, il est acheminé par le réseau à toutes les interfaces membres de ce groupe. Il est à noter qu'il n'existe plus d'adresses de type broadcast comme sous IPv4. Elles sont remplacées par des adresses de type multicast.

Le dernier type d'adresses est le type anycast. Comme dans le cas du type multicast, une adresse de type anycast désigne un groupe d'interfaces. La différence étant que lorsqu'un paquet a pour destination une telle adresse, il est acheminé à un des éléments du groupe et non pas à tous. Le paquet est par exemple acheminé à l'interface la plus proche au sens de la métrique des protocoles de routage.

La répartition des adresses est décrite dans le RFC 2373. Le tableau en annexe 1 le résumé du fractionnement de l'espace d'adressage. Les premiers bits de l'adresse –le préfixe– définissent son type. Les adresses commençant par 8 zéros sont réservées, notamment pour les adresses spéciales - adresse indéterminée, de bouclage, mappée, compatible -. C'est ainsi que toutes les adresses commençant par 80 zéros sont réservées aux adresses IPv4. Deux variantes sont supportées. Elles se distinguent selon les 16 bits suivants –soient 16 bits à 0 ou à 1–. Ces variantes décrivent notamment comment les datagrammes IPv6 doivent être encapsulés pour traverser en mode tunnel l'actuelle infrastructure IPv4.

On notera que plus de 70% de l'espace disponible n'a pas été alloué, ce qui permet de conserver toute latitude pour l'avenir.

### 1.7.5 Plan d'adressage agrégé –Aggregatable Global Unicast Address Format–

Plusieurs plans d'adressages ont été étudiés pour arriver finalement au plan d'adressage dit agrégé. Ce plan d'adressage proposé par l'IETF est défini dans le RFC 2374. Ce format d'adresse est conçu pour supporter l'agrégation actuelle des adresses au niveau des fournisseurs de services et le nouveau type d'agrégation. La combinaison des deux va permettre un routage efficace des datagrammes.

Ce plan d'adressage –figure 1.6– comprend trois niveaux de hiérarchie:

- une topologie publique –48 bits–;
- une topologie de site –16 bits–;
- un identifiant d'interface –64 bits–.

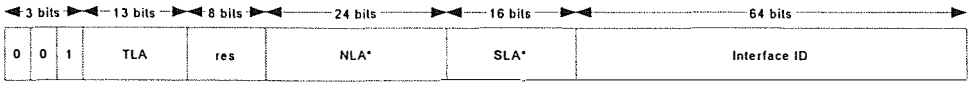



**Fig. 1.6** - Format des adresses de type plan agrégé

La topologie publique est constituée par l'ensemble des prestataires et par les points d'échange de connectivité IP. Le format de la topologie publique est le suivant:

- Un préfixe 2000::/3 identifiant le plan d'adressage.
- Une unité d'agrégation haute –TLA ou Top Level Aggregator– sur 13 bits représente les grands opérateurs intercontinentaux. Ces opérateurs gèrent généralement un réseau où les tables de routage ne contiennent aucune route par défaut. Ces préfixes ne sont attribués qu'aux opérateurs offrant un trafic de transit.
- Une partie réservée sur 8 bits permet de faire évoluer le plan d'adressage. En effet il est difficile de prévoir les besoins en TLA et en NLA. La taille actuelle du champ permet de contenir jusqu'à 8 192 identifiants de TLA. Si dans le futur, la taille de ce champ est insuffisante, des bits seront pris dans le champ réservé. De même si dans le futur, le nombre de bits est insuffisant pour numéroter les NLA, des bits pourront être pris à droite du champ réservé. Quand tous les bits du champ réservé seront pris, un préfixe autre que 2000::/3 pourra être attribué.
- des unités d'agrégation basse –NLA ou Next Level Aggregator– dont la longueur totale est de 24 bits. La dernière partie d'agrégation basse constitue l'identificateur de site -ou domaine-. Le choix du découpage en unités d'agrégation basse est laissé à la discrétion de chaque unité d'agrégation haute. De cette façon nous pouvons avoir une succession de plusieurs NLA. Ceux-ci représentent les opérateurs intermédiaires échangeant leur connectivité en des points d'interconnexion avec les TLA.

La topologie du site –SLA: Site Level Agregator– codé sur 16 bits est sur la responsabilité du gestionnaire du site. Celui-ci peut hiérarchiser son plan d'adressage, ce qui implique la présence du caractère "*" sur le schéma de la figure 1.4.

L'identifiant d'interface *interface ID* permet de distinguer les interfaces connectées sur un même lien. Il est codé sur 64 bits et possède un format bien défini. Un bit -le septième à partir du bit de poids fort- est réservé pour indiquer l'unicité mondiale de l'interface s'il est à 1.

Il existe plusieurs méthodes pour construire l'identifiant d'une interface. Si une interface possède un identificateur global IEEE EUI-64, celui-ci a la structure décrite à la figure 1.7.

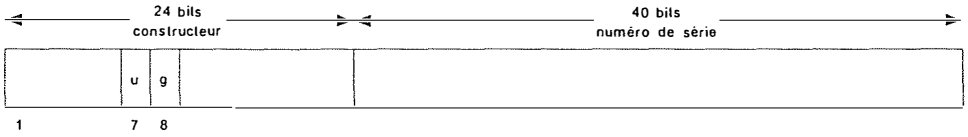



Fig. 1.7 - Identificateur global IEEE à 64 bits EUI-64

Les 24 premiers bits de EUI-64, comme pour les adresses MAC IEEE 802 identifient le constructeur et les 40 derniers identifient le numéro de série. Les deux bits *u* et *g* respectivement le septième et le huitième du premier octet ont une signification particulière. *u* –Universal– vaut 0 si l'identifiant EUI-64 est universel. *g* –Groupe– indique si l'adresse est individuelle –*g* = 0–, c'est-à-dire désigne un seul équipement sur le réseau ou de groupe –g = 1–. C'est, par exemple, le cas d'une adresse multicast.

Pour la construction des adresses IPv6, il a été préféré d'utiliser *u* = 1 pour marquer l'unicité mondiale.

Si une interface possède une adresse MAC IEEE 802 à 48 bits universelle -cas des interfaces Ethernet-, cette adresse est utilisée pour construire des identifiants d'interface sur 64 bits comme indiqué à la figure 1.8.

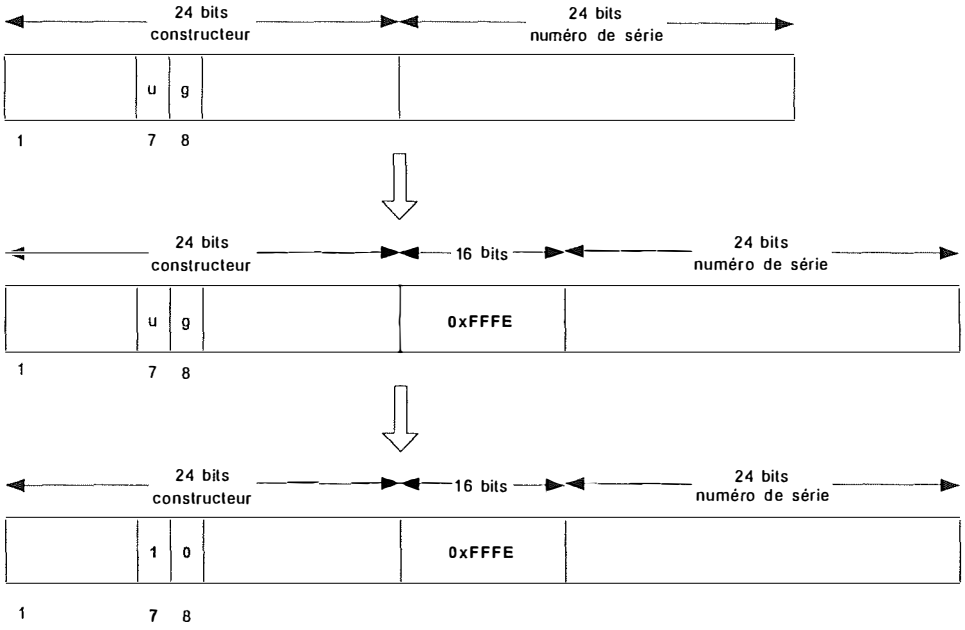



Fig. 1.8 - Transformation d'uneadresse MAC en identifiant d'interface

Si une interface possède une adresse locale unique sur le lien mais non universelle -par exemple format d'adresse IEEE 802 sur 2 octets-, l'identifiant d'interface est construit à partir de cette adresse en rajoutant des 0 en tête pour atteindre 64 bits.

Si une interface ne possède aucune adresse –comme c'est le cas pour des interfaces utilisées pour des liaisons PPP– et si la machine n'a pas d'identifiant EUI-64, il n'y a pas de méthode unique pour créer un identifiant d'interface. Il est toutefois conseillé d'utiliser l'identifiant d'une autre interface si c'est possible ou de le générer aléatoirement avec le bit *u* positionné à 0. S'il y a conflit –les deux extrémités ont choisi la même valeur–, celui-ci est détecté lors de

l'initialisation de l'adresse lien-local de l'interface. Dans ce cas, le conflit d'identifiant devra être réglé manuellement.

### 1.7.6 Autres types d'adresses unicast

#### 1.7.6.1 Adresse indéterminée –unspecified address–

L'adresse indéterminée est utilisée comme adresse source par un nœud du réseau pendant son initialisation, avant d'acquérir une adresse. Sa valeur est 0:0:0:0:0:0:0:0 en abrégé :: .

Cette adresse est utilisée uniquement par les protocoles d'initialisation. Elle ne doit jamais être attribuée à un nœud et ne doit jamais apparaître comme adresse de destination d'un paquet IPv6.

#### 1.7.6.2 Adresse de bouclage –loopback address–

L'adresse de bouclage vaut 0:0:0:0:0:0:0:1 en abrégé ::1. C'est l'équivalent de l'adresse 127.0.0.1 d'IPv4. Elle est utilisée par un nœud pour s'envoyer à lui-même un paquet IPv6.

Un paquet IPv6 transitant sur le réseau ne peut avoir l'adresse de bouclage comme adresse source ni comme adresse de destination.

#### 1.7.6.3 Adresses lien-local -link local address-

Les adresses de type lien-local sont des adresses dont la validité est restreinte à un lien, c'est-à-dire l'ensemble des interfaces directement connectées sans routeur intermédiaire. C'est par exemple le cas des équipements branchés sur un même Ethernet ou des équipements reliés par une connexion PPP. Les adresses lien-local sont configurées automatiquement à l'initialisation de l'interface et permettent la communication entre nœuds voisins. Elle sont obtenues en concatenant le préfixe FE80::/64 aux 64 bits de l'identifiant de l'interface. Le format de telles adresses est représenté à la figure 1.9.

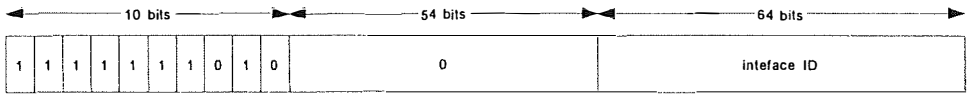



**Fig. 1.9** - Adresse lien-local

Ces adresses sont utilisées par les protocoles de configuration d'adresse globale, les protocoles de découverte des voisins –Neighbor Discovery– et de découverte des routeurs –Router Discovery–. Ces protocoles font parti des nouveaux dispositifs de IPv6 par rapport à IPv4. Le premier –découverte des voisins– remplaçant en particulier le protocole ARP–Address Resolution Protocol–défini dans le RFC 826. Ces dispositifs permettent à un réseau local de se configurer automatiquement.

Les adresses lien-local sont uniques à l'intérieur d'un lien. Le protocole de détection de duplication d'adresse permet de s'en assurer. Par contre la duplication d'une adresse lien-local entre deux liens différents ou entre deux interfaces d'un même nœud est autorisée.

Un routeur ne doit en aucun cas retransmettre un paquet ayant pour adresse source une adresse de type lien-local.

### 1.7.6.4 Adresses site-local

Les adresses site-local sont des adresses dont la validité est restreinte à un site. Par exemple un site qui n'est pas encore connecté à l'Internet peut utiliser ces adresses ; ce qui le dispensera de demander ou d'emprunter un préfixe. Ce système généralise le concept d'adresse privée d'IPv4.

Une adresse site-local est obtenue en concatenant au préfixe FEC0::/48, un champ de 16 bits qui permet de définir plusieurs sous-réseaux et les 64 bits de l'identifiant de l'interface. Son format est représenté à la figure 1.10.

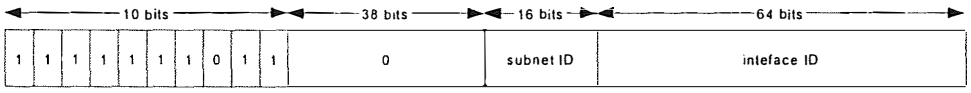



**Fig. 1.10** - Adresse site-local

Si le site est relié à l'Internet, ces adresses doivent être configurées à l'intérieur du site par les routeurs qui permettent la sortie du site.

L'usage de ces adresses pour un site connecté à l'Internet est encore mal compris et soulève de nombreuses questions [Hui96]:

- quelle est la définition exacte d'un site?
- le plan d'adressage de site –le champ *subnet ID*– est-il le même pour les adresses globales –le champ SLA de l'adresse agrégée– et les adresses de site?
- une machine peut-elle appartenir à deux sites différents?
- comment savoir si un partenaire est dans le site? Comment choisir les bonnes adresses source et destination? Quelles adresses doit renvoyer un serveur DNS?

Par contre un des avantages des adresses site-local est quelles ne sont pas modifiées lors du changement de fournisseur de connectivité. Ce qui ne perturbe donc pas les communications locales.

### 1.7.6.5 Les adresses IPv4 mappées

Elles sont représentées sous la forme ::FFFF:a.b.c.d où a.b.c.d est une adresse IPv4 –voir figure 1.11–.

Ces adresses permettent à une machine de communiquer en IPv4 avec une machine IPv4 tout en restant dans la famille des adresses IPv6. Pour qu'un serveur puisse répondre à des requêtes IPv6 et IPv4, il lui faut une double pile de communication. En émission, la machine voyant une adresse IPv4 mappée utilise la pile de communication IPv4 et envoie des paquets IPv4. En réception, une requête IPv4 est reçue par la pile IPv4 puis présentée aux applications sous la forme d'une requête IPv6 comportant des adresses de type IPv4 mappées. Une adresse mappée n'apparaît jamais sous cette forme sur le réseau.


**Fig. 1.11** - Adresse IPv4 mappée

### 1.7.6.6 Les adresses IPv4 compatibles

Elles sont représentées sous la forme ::a.b.c.d où a.b.c.d est une adresse IPv4 –voir figure 1.12–.

Ces adresses permettent à deux machines IPv6 de communiquer entre elles en IPv6 à travers un tunnel IPv6/IPv4. Un paquet IPv6 transmis vers l'adresse ::a.b.c.d est encapsulé dans un paquet IPv4 qui est acheminé à travers le réseau IPv4 vers l'adresse a.b.c.d. Au niveau de la machine destinatrice, le paquet IPv6 est extrait et traité normalement par la pile de communication IPv6.


**Fig. 1.12** - Adresse IPv4 compatible

### 1.7.6.7 Adresses multicast

Une adresse de type multicast désigne un ensemble d'interfaces. Elle est caractérisée par le préfixe FF00::/8. Son format est donné à la figure 1.13.

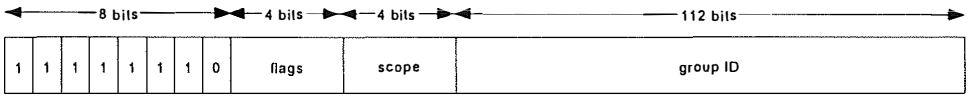



**Fig. 1.13** - Adresse de multicast

Le champ *flags* est un mot de 4 bits dont les trois premiers sont réservés et doivent être initialisés à zéro. Le dernier bit nommé T, s'il est positionné à 0, indique que la validité de l'adresse est permanente donc attribuée par une autorité compétente de l'Internet, sinon l'adresse est temporaire.

Le champ suivant, également de longueur 4 bits est le niveau de diffusion -scope- de l'adresse considérée. Les valeurs actuellement définies sont:

0 réservé

1 nœud –node local scope–

2 lien –link local scope–

5 site –site local scope–

8 organisation –organisation local scope–

E global –global scope–

F réservé

L'intérêt du niveau de diffusion est de garantir le confinement des paquets dans une zone déterminée. Cette méthode est plus rigide mais plus sûre qu'en IPv4 où la portée d'un paquet de multicast est définie en fonction du champ TTL.

Il faut noter qu'une adresse temporaire n'est valide qu'au sein de son niveau de diffusion.

Le dernier champ identifie le groupe –group ID– dans lequel est limitée la diffusion des paquets issus d'une machine appartenant au même groupe.

Un paquet IPv6 ne peut avoir pour adresse source une adresse de type multicast.

#### 1.7.6.8 Adresses multicast prédéfinies

Les adresses multicast prédéfinies ont un champ *flags* valant 0 -adresse permanente-. A chaque protocole capable utiliser la diffusion correspond un groupe particulier. De cette façon chaque protocole de routage a son propre groupe. Pour chaque groupe, seules certaines valeurs de champ *scope* sont autorisées.

Le groupe 0 est réservé. Ce sont les seize adresses FF0X:: où X varie de 0 à F. Plusieurs groupes ont été définis.

- Le groupe 1 de tous les nœuds IPv6 défini seulement au niveau de diffusion 1 et 2. Par exemple l'adresse FF02::1 désigne tous les nœuds IPv6 sur le même lien-local que l'interface de l'expéditeur.
- Le groupe 2 de tous les routeurs IPv6 défini au niveau de diffusion 1, 2 et 5. De cette façon l'adresse FF05::2 désigne tous les routeurs IPv6 du site.
- Le groupe 9 de tous les routeurs utilisant le protocole de routage RIP –défini au niveau de diffusion 2
- Le groupe 0x10002 de tous les agents DHCP –Dynamic Host Configuration Protocol, le groupe 0x10003 de tous les serveurs DHCP et le groupe 0x10004 de tous les relais DHCP

#### 1.7.6.9 Adresses multicast sollicité –solicited node address–

Ce type d'adresse est construit à partir d'une adresse unicast ou d'une adresse anycast en concaténant le préfixe FF02::1:FF00:0/104 aux 24 derniers bits extraits de celle-ci.

Un équipement, à partir de chacune de ses adresses IPv6 unicast et anycast construit une adresse de multicast sollicité et écoute les paquets émis vers cette adresse. Les autres stations sur le lien connaissant l'adresse IPv6 d'un équipement mais ignorant son adresse MAC peuvent utiliser l'adresse de mulicast sollicité pour le joindre.

Les adresses de multicast sollicité sont entre autres utilisées par le protocole ICMPv6 – détection des adresses dupliquées, découverte des voisins–.

L'utilisation des adresses multicast sollicité rendent obsolète l'utilisation de la diffusion généralisée qu'utilise le protocole ARP en IPv4.

#### 1.7.6.10 Adresses anycast

Au lieu d'envoyer un paquet à une interface déterminée, on envoie ce paquet à une adresse représentant un groupe d'interfaces (généralement attachées à des nœuds différents), à charge

pour le système de routage de l'acheminer vers l'interface la plus proche au sens de la métrique des protocoles de routage. Si plusieurs interfaces associées à une adresse anycast répondent à une sollicitation, seule l'une d'entre elles sera utilisée durant le reste de la session.

Les adresses de type anycast ont un format identique à celui des adresses de type unicast. Elles sont allouées dans le même espace d'adressage sans restriction aucune. On crée une adresse anycast en attribuant une même adresse unicast à des nœuds distincts, chacun des nœuds devant être configuré pour que l'adresse ainsi attribuée soit de type anycast (sinon on aurait une adresse unicast dupliquée.

Une adresse de type anycast ne peut être attribuée qu'à un routeur. Actuellement seule l'adresse anycast d'un sous-réseau est définie. Elle est fabriquée en concatenant le préfixe correspondant au sous-réseau et un suffixe nul -Figure 1.14-. Un paquet à destination de l'adresse anycast d'un sous-réseau sera acheminé à un routeurs IPv6 ou à une station IPv6 appartenant à ce sous-réseau.

Un paquet IPv6 ne peut avoir pour adresse source une adresse de type anycast.

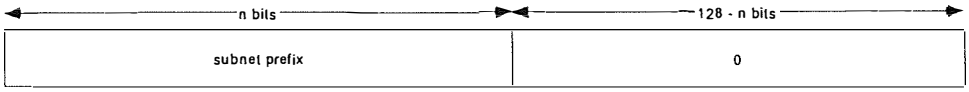



Fig. 1.14 - Adresse anycast "sous-réseau"

## 1.8 Extensions

Dans la nouvelle version du protocole IP, quelques champs manquants du datagramme IPv4 sont néanmoins dans certaines circonstances requis. C'est ainsi qu'IPv6 a introduit le concept d'en-tête d'extension –optionnel–. Cet en-tête, s'il est présent, fournit une information complémentaire de façon efficace. Six types d'en-tête d'extension ont été définis dans un premier temps, ils sont listés à l'annexe 2.

En IPv4, si certains routeurs ne peuvent pas traiter le trafic multicast, celui-ci est encapsulé dans un tunnel point-à-point pour traverser les routeurs intermédiaires. Cette liaison est appelée un tunnel IPv4. Les extensions d'IPv6 peuvent être considérées comme un prolongement de l'encapsulation. Mises à part les extensions de proche-en-proche et de routage traitées par les routeurs intermédiaires, les autres extensions ne sont prises en compte que par les équipements destinataires du paquet.

Une extension a une longueur multiple de 8 octets. Certains en-têtes ont un format fixe, d'autres contiennent un nombre variable de champs variables. Chaque item est codé sous forme d'un triplet (Type, Longueur, Valeur).

*Type* est un champ d'un octet qui définit le type de l'option. Les différents types ont été choisis de façon à ce que les deux premiers bits disent quoi faire aux routeurs qui ne savent pas exécuter l'option.

*Longueur* est un champ d'un octet qui indique la taille du champ *Valeur* -de 0 à 255 octets- qui contient une information quelconque adressée au destinataire.

Si plus d'un en-tête est présent dans le datagramme, ils doivent apparaître immédiatement après l'en-tête fixe, de préférence dans l'ordre recommandé par le RFC 2460, à savoir proche-en-proche, destination, routage par la source, fragmentation, authentification et destination.

### 1.8.1 Proche-en-proche

Cette extension contient des informations destinées à tous les routeurs sur le chemin. Elle est toujours située en première position. Bien que plusieurs options soient prévues pour cet en-tête, une est actuellement définie, à savoir permettre l'utilisation des datagrammes d'une taille supérieure à 64 Ko. Le format de cette extension est défini à la figure 1.16.

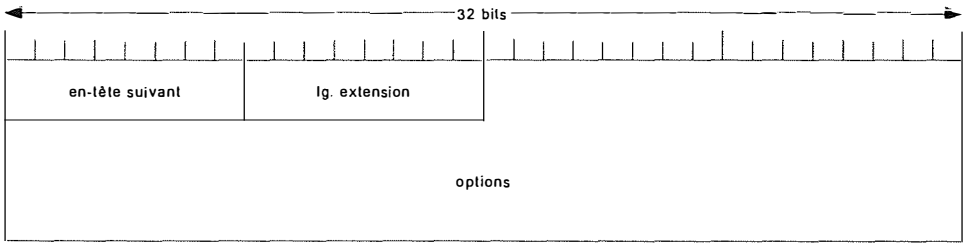



Fig. 1.16 - Format de l'extension de proche-en-proche

Le type associé -contenu dans le champ *en-tête suivant* de l'en-tête précédent- est 0 et le champ *Longueur de l'extension* contient le nombre de mots de 64 bits moins 1.

L'extension est composée d'options. Quatre options dont deux de bourrage sont définies –voir figure 1.17–. Chaque option est une suite d'octets. Le premier code le type de l'option. Ses deux premiers bits de poids fort définissent le comportement du routeur quand il rencontre une option inconnue comme le montre l'annexe 3.

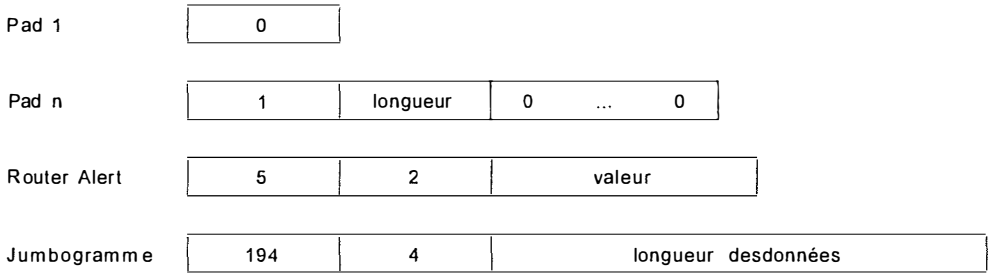



**Fig. 1.17** - Format des options IPv6

### 1.8.2 Destination

Cette extension dont le format est identique à l'extension de proche-en-proche contient des options qui sont traitées par l'équipement destinataire. Cette option permet de limiter le niveau d'encapsulation dans les tunnels des paquets IPv6.

### 1.8.3 Routage

L'en-tête routage donne la liste d'un ou de plusieurs routeurs qui doivent être visités sur le trajet vers la destination. Cette extension permet d'imposer à un paquet une route différente de

celle offerte par les politiques de routage présentes sur le réseau. Pour l'instant, seul le routage par la source –type = 0–, similaire à l'option Loose Source Routing d'IPv4 est défini.

Dans IPv4, le routage peut être strict –le routeur suivant présent dans la liste doit être un voisin directement accessible– ou libéral –loose, un routeur peut utiliser les tables de routage pour joindre le routeur suivant servant de relais–. Dans IPv6, seul le routage libéral est autorisé. En effet le routage mixte était mis en place pour des raisons de sécurité. La source devait être sûre du chemin pris par les paquets. Cette utilisation a maintenant disparu du réseau.

Le principe du routage par la source ou Source Routing d'IPv4 est le même pour IPv6. L'émetteur met dans le champ destination du paquet IPv6 l'adresse du premier routeur servant de relais. L'extension contient la suite de la liste des autres routeurs relais et destinataires. Quand un routeur reçoit un paquet qui lui est adressé comportant une extension de routage par la source, il permute son adresse avec l'adresse du prochain routeur et réémet le paquet vers cette adresse.

La figure 1.18 donne le format de l'extension du routage par la source.

Le champ *longueur de l'extension* indique le nombre de mots de 64 bits qui composent celle-ci. Pour l'extension de type 0, cela correspond au nombre d'adresses présentes dans la liste, multiplié par 2.

Le champ *type de routage* indique la nature du routage -routage par la source-.

Le nombre de segments restants est décrémenté après la traversée d'un routeur. Il indique le nombre d'équipements qui doivent encore être traversés. Il permet de trouver l'adresse qui devra être substituée.

Les 32 bits suivants son inutilisés afin de préserver l'alignement.

La liste comprenant les routeurs à traverser et destinataire est fournie. Ces adresses ne peuvent pas être multicast.

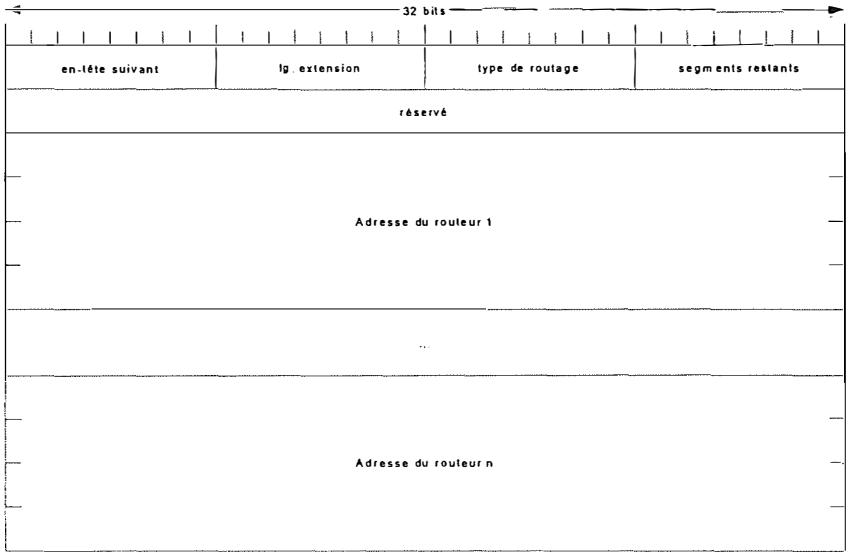



Fig. 1.18 - Format de l'extension routage par le source

### 1.8.4 Fragmentation

La fragmentation telle qu'elle est pratiquée dans IPv4 n'est pas très performante. En effet, quand un routeur ne peut pas transmettre un paquet à cause de sa grande taille et si le bit *DF* – Don't Fragment– est à 0, il découpe l'information à transmettre en fragments. Or, le réseau IP étant un réseau à datagrammes, il n'y a pas de possibilité de contrôler les fragments. De cette façon deux fragments successifs peuvent prendre deux chemins différents et par conséquent, seul le destinataire peut effectuer l'assemblage. Si un fragment se perd, le réassemblage est purement et simplement impossible.

Il est plus intéressent d'adapter la taille des paquets à l'émission. Ceci est fait en utilisant les techniques de découverte du MTU -Maximun Transmission Unit-. En pratique, une taille de paquet de 1500 octets est presque universelle.

Le format de l'extension de fragmentation est donné à la figure 1.19. La signification des champs est identique à celle de IPv4.

Le champ *place du fragment* indique lors du ré assemblage où les données doivent être insérées. Ce qui permet de résoudre les problèmes dus au déséquencement dans les réseaux orientés datagrammes.

Le bit *M* s'il vaut 1 indique qu'il y aura d'autres fragments émis.

Le champ *identification* permet de repérer les fragments appartenant à un même paquet initial. Il est différent pour chaque paquet et est recopié dans ses fragments. Ce champ est l'équivalent du champ *identification* de l'en-tête de base des datagrammes IPv4.

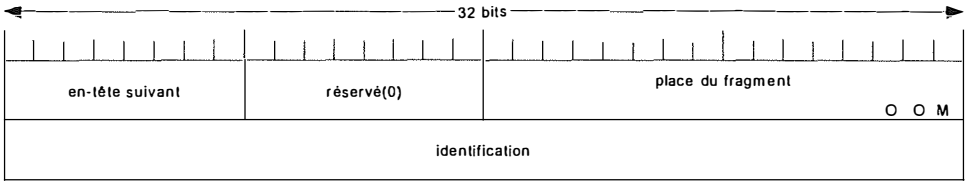



**Fig. 1.19** - Format de l'extension de fragmentation

Le bit *DF* utilisé en IPv4 n'est plus nécessaire, car si l'extension n'est pas présente, le routeur rejette le paquet.

Dans IPv4, la valeur d'une option était codée de manière à indiquer au routeur effectuant la fragmentation si elle devait être copiée dans le fragment. Dans IPv6, l'en-tête et les extensions qui concernent les routeurs intermédiaires sont recopiées dans chaque fragment.

### 1.8.5 Sécurité

Deux extensions de sécurité sont définies par l'IETF : les extensions d'authentification AH – Authentication Header– et de confidentialité ESP –Encapsulating Security Payload–. Elles permettent de protéger les communications passées sur les réseaux IPv6 mais aussi IPv4 en assurant les services de confidentialité, authentification et intégrité.

## 1.9 Les autres champs de l'en-tête du datagramme IPv4

Dans cette section, nous présentons les différents champs de l'en-tête du datagramme IPv4 dont l'équivalent n'existe pas dans l'en-tête des datagrammes de la nouvelle version IPv6. Parmi ceux-ci nous avons les champs *Checksum*, *Identification*, *Drap* et *Dep_fragment*. Les trois derniers champs traitent les problèmes liés à la fragmentation des paquets, tandis que le premier sert au contrôle des erreurs au niveau de l'en-tête.

### 1.9.1 Les champs de fragmentation d'IPv4

Le champ *identification* permet à l'ordinateur destinataire de déterminer à quel datagramme appartient le fragment reçu. Tous les fragments d'un même datagramme contiennent la même valeur d'identification.

Le champ *Drap* -Drapeau- couvre 3 bits dont un est inutilisé. Il comprend deux indicateurs: *DF* qui signifie "ne pas fragmenter" et *MF* qui signifie "fragment non unique" –more fragment–. DF indique au routeur de ne pas fragmenter les datagrammes car le destinataire serait incapable de le reconstituer. En marquant le datagramme avec le bit *DF*, l'émetteur sait qu'il parviendra en un seul tenant, même si le datagramme doit renoncer au cheminement optimal. Tous les équipements doivent être capables d'accepter des fragments de 576 octets au moins.

Dans le cas de datagrammes fragmentés, tous les fragments sauf le dernier ont leur indicateur *MF* activé. Cela permet de déterminer quand tous les fragments d'un datagramme sont arrivés.

Le champ *Dép_fragment* précise la localisation ou le déplacement du fragment dans le datagramme courant. Tous les fragments, excepté le dernier du datagramme, doivent avoir des longueurs multiples de 8 octets, l'unité élémentaire du fragment. Comme le champ contient 13 bits, il y a au maximum 8192 fragments par datagramme, ce qui correspond à la longueur maximale de 65536 octets, soit un octet de plus que la valeur du champ *Longueur totale.*

En IPv6, la fragmentation des paquets n'est plus gérée au niveau de l'en-tête de base du datagramme, elle est reléguée à l'extension de fragmentation. La différence fondamentale est que seule la source peut réduire un datagramme en plusieurs fragments. D'où l'inutilité du bit *DF* dans la nouvelle version du protocole. A part cette exception, toute la sémantique contenue dans les champs relatifs à la fragmentation de l'en-tête des datagrammes IPv4 a été reprise dans l'extension de fragmentation d'IPv6.

### 1.9.2 Checksum

En IPv4, le champ checksum vérifie exclusivement la validité de l'en-tête. Sa fonction est la détection d'erreur à l'intérieur des routeurs. Il est à noter que le total de contrôle doit être recalculé à chaque saut puisqu'un champ au minimum aura changé entre temps : la durée de vie, qui a été décrémentée d'une unité. Cette vérification étant faite à chaque retransmission par un routeur, ce qui entraîne une augmentation du temps de traitement des paquets.

Ce champ a été éliminé dans IPv6. Ce qui pourrait à première vue apparaître comme une erreur. Mais il a été montré que les erreurs traquées par le checksum de l'en-tête IP sont peu fréquentes car les support physiques sont de meilleure qualité et arrivent à détecter des erreurs- cas du CRC dans Ethernet-. Les erreurs apparaissent surtout au niveau des routeurs

lors du calcul de cette checksum. Donc une erreur pourrait subvenir alors que le checksum est correct.

Le checksum sur l'en-tête IPv6 n'existant plus, il faut quand même se prémunir des erreurs de transmission. En particulier, une erreur sur l'adresse de destination va faire router un paquet dans une mauvaise direction. Le destinataire doit donc vérifier que les informations d'en-tête IP sont incorrectes pour éliminer ces paquets. Dans les mises en œuvre des piles de protocole Internet, les entités de niveau transport remplissent certains champs de niveau réseau. Il a donc été décidé que tous les protocoles au-dessus d'IPv6 devaient utiliser une somme de contrôle intégrant à la fois les données et les informations de l'en-tête IPv6. Pour un protocole comme TCP qui possède une somme de contrôle, cela signifie modifier le calcul de cette somme. Pour un protocole comme UDP qui possède une somme de contrôle facultative, cela signifie modifier le calcul de cette somme et le rendre obligatoire.

IPv6 calcule le checksum à partir de la concaténation d'un pseudo-en-tête -composé des champs: adresse source, adresse destination, longueur des données, en-tête suivant- et du paquet du protocole concerné.

## 1.10 Le protocole de contrôle ICMPv6 -RFC 2463-

Le nouveau protocole IP a revu le protocole de contrôle ICMP –Internet Control Message Protocol–. En effet dans IPv4, ICMP sert à la détection d'erreurs -par exemple durée de vie expirée- au test –par exemple demande d'écho par la commande ping– et à la configuration automatique des équipements –par exemple découverte des routeurs–. Ces trois fonctions sont mieux définies dans IPv6. En outre, ICMPv6 intègre les fonctions de gestion des groupes de multicast qui sont effectuées par IGMP –Internet Group Management Protocol– dans IPv4. ICMPv6 reprend également les fonctions du protocole ARP utilisé par IPv4.

Le format général des messages ICMP est donné à la figure 1.20. Ces messages peuvent être groupés en deux: les messages d'erreur et les messages d'information. Les messages d'erreur sont identifiés par la présence de 0 au niveau du bit de poids fort du champ type. Cela implique que le type des messages d'erreur varie entre 0 et 127, tandis que les valeurs entre 128 et 255 sont réservées aux messages d'information.

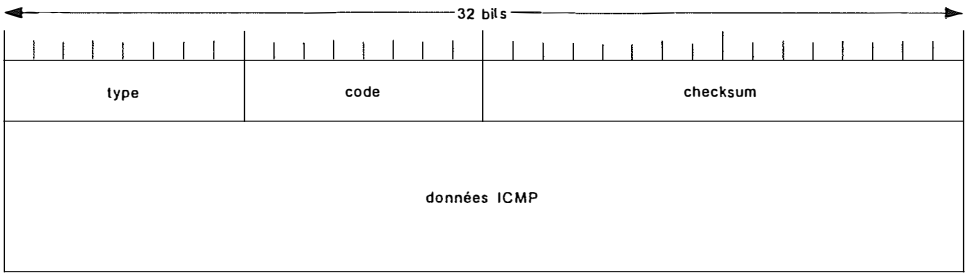



Fig 1.20 - Format générique d'un message ICMPv6

Le champ *type* représente la nature du message ICMP. Contrairement à IPv4 où la numérotation ne suivait aucune logique, nous avons un intervalle précis de valeurs pour les messages d'erreur et les messages d'information.

Le champ *code* précise la cause du message ICMPv6.

Le champ *checksum* permet de vérifier l'intégrité du paquet ICMPv6 et les parties de l'en-tête IPv6.

Les messages ICMPv6 de compte rendu d'erreur contiennent dans la partie donnée le paquet IPv6 qui a provoqué l'erreur.

L'anne4 reprend les différents type de messages ICMPv6 et les codes correspondants.

### 1.10.1 Destination inaccessible

Ce message est émis par un routeur ou par la couche IPv6 d'un nœud lorsqu'un paquet ne peut être délivré à sa destination pour des raisons autres que la congestion. Ce message peut être généré pour diverses raisons :

- le routeur ne trouve pas dans ses tables la route vers la destination –code = 0–;
- le franchissement d'un équipement de type firewall est interdit pour des raisons administratives par exemple –code = 1–;
- toute autre raison d'inaccessibilité comme par exemple la tentative de router une adresse locale au lien –code = 3–;
- le port de destination contenu dans le paquet n'est pas affecté à une application –code = 4–.

Le format du message ICMPv6 envoyé pour l'inaccessibilité d'une destination est représenté dans la figure 1.22.

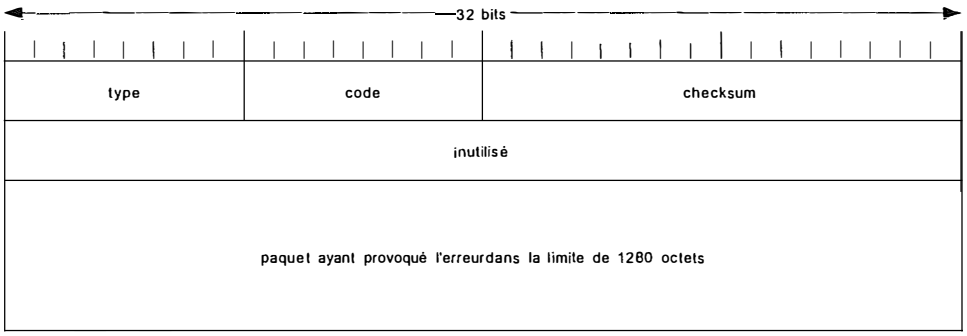



Fig. 1.22 - Format du message ICMPv6 destination inaccessible

### 1.10.2 Paquet trop grand

Ce message est utilisé par le protocole de découverte du MTU pour trouver la taille optimale des paquets IPv6 afin que ceux-ci puissent traverser les routeurs sur la route. Il contient en fait la taille du MTU acceptée par le routeur pour que la source puisse adapter la taille des données.

Cette information manquait cruellement dans les spécifications initiales de IPv4. Ce qui compliquait la découverte de la taille maximale des paquets utilisables sur l'ensemble du chemin. Mais le RFC 1191 proposait déjà une modification qui inclurait cette information.

Le format du message ICMPv6 envoyé lorsque la taille des paquets est trop grande est représenté à la figure 1.23.

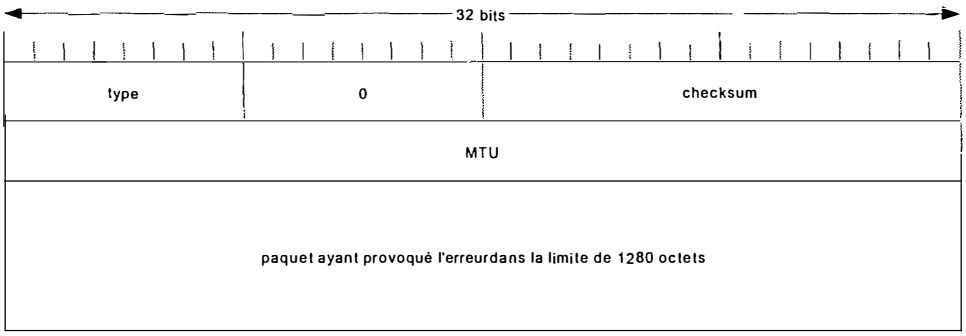



**Fig. 1.23** - Format du message ICMPv6 paquet trop grand

### 1.10.3 Temps dépassé

Ce message indique que le paquet a été rejeté par le routeur soit parce que le champ nombre de sauts a atteint 0 –code = 0– soit qu'un fragment s'est perdu et que le temps alloué au ré assemblage a été dépassé –code = 1–. Il sert aussi à la commande *traceroute* à déterminer le chemin pris par les paquets. Son format est représenté à la figure 1.24.

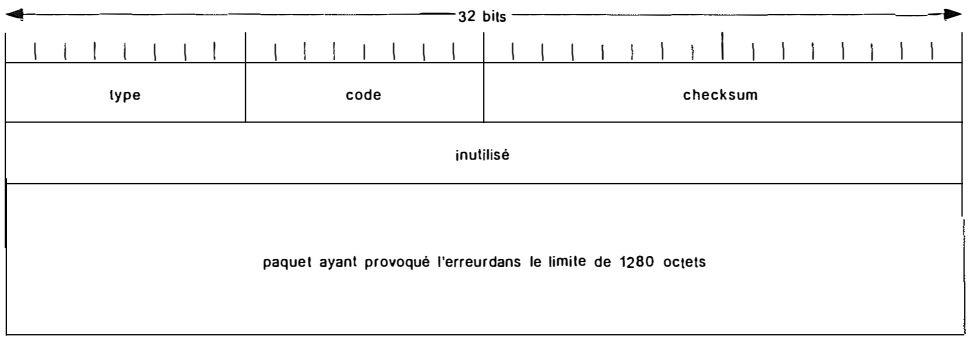



**Fig. 1.24** - Format du message ICMPv6 temps dépassé

### 1.10.4 Erreur de paramètre

Ce message est émis par un nœud quand celui-ci détecte une erreur de syntaxe dans l'en-tête du paquet IP ou dans les extensions. Le champ *code* révèle la cause de l'erreur.

Code = 0 si la syntaxe de l'en-tête n'est pas correcte.

Code = 1 si le numéro en-tête suivant n'est pas reconnu.

Code = 2 si une option de l'extension n'est pas reconnue et le codage des bits de poids fort oblige à rejeter le paquet.

Le champ pointeur indique l'octet où l'erreur est survenue dans le paquet retourné. Le format du message ICMPv6 correspondant est représenté à la figure 1.25.

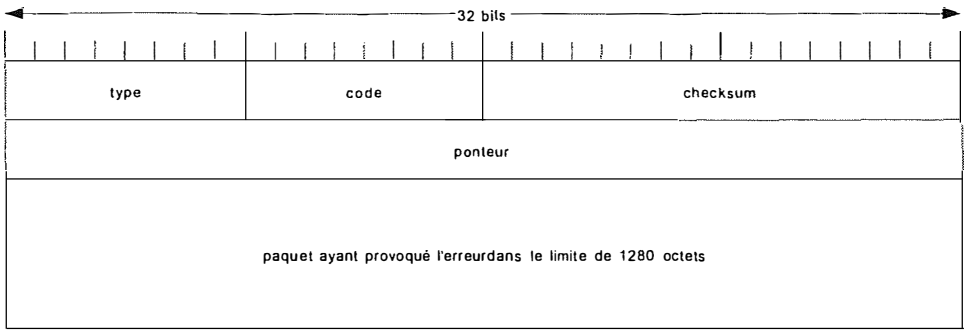



Fig. 1.25 - Format du message ICMPv6 erreur de paramètre

### 1.10.5 Demande et réponse d'écho

Ces deux messages servent en particulier à la commande *ping* permettant de tester l'accessibilité d'une machine. Le principe de fonctionnement est le même que pour IPv4. Une requête de type 128 est envoyé vers l'équipement dont on veut tester le fonctionnement. Celui-ci répond par le message *réponse d'écho* de type 129.

Le champ *identificateur* permet de distinguer les réponses dans le cas où plusieurs commandes *ping* seraient lancées simultanément sur la machine.

Le champ *numéro de séquence* permet d'associer la réponse à une requête pour mesurer le temps d'aller et de retour dans le cas où les demandes sont émises en continu et que le délai de propagation est élevé.

Le champ *données* permet d'augmenter la taille du message pour les mesures.

Le format du message est représenté à la figure 1.26.

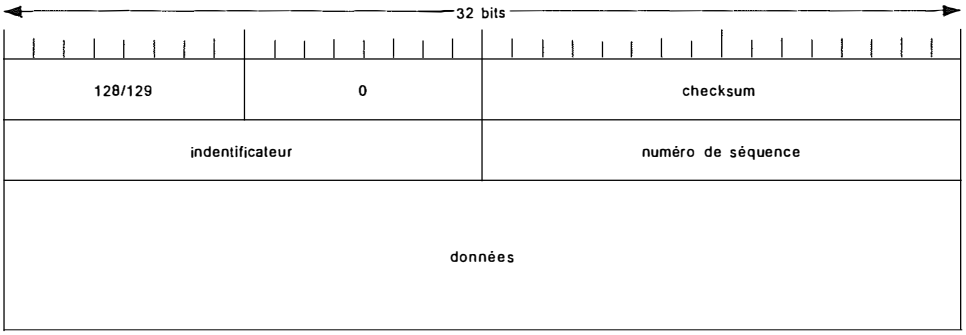



Fig. 1.26 - Format du message ICMPv6 demande et réponse d'écho

### 1.10.6 Gestion des groupes multicast

La gestion des groupes multicast est basée sur trois types de messages. Le premier message est celui de demande des équipements participant aux groupes multicast. Il est émis périodiquement par les routeurs qui gèrent la diffusion en multicast pour déterminer si la diffusion intéresse toujours au moins un équipement sur le support. Le deuxième message, celui du rapport d'abonnement est émis par un équipement abonné à un groupe multicast. Il est émis quand un équipement s'abonne à un groupe ou en réponse au message précédent. Le

troisième message est celui de réduction d'un groupe de multicast. Il est émis quand le dernier équipement quitte un groupe multicast.

Les paquets IPv6 qui transportent ces messages ont obligatoirement l'option Router Alert. Suivant la nature du message, le champ adresse de destination peut contenir:

- l'adresse de tous les routeurs du lien local –FF02::2– dans un message de réduction d'un groupe de multicast ;

- l'adresse du groupe en question dans un message de rapport d'abonnement ;

- l'adresse multicast de ce groupe s'il s'agit d'un groupe spécifique sinon l'adresse générale de diffusion sur le lien –FF::1– pour connaître l'état de tous les groupes.

Le champ nombre de sauts vaut 1.

Les champs du message ICMPv6 contiennent :

- Type = 130 pour le recensement de groupes multicast ;

  Type = 131 pour le rapport d'abonnement ;

  Type = 132 pour la réduction d'un groupe multicast ;

- Code = 0 ;

- Délai maximal de réponse = retard maximal autorisé –en millisecondes– des messages de rapport d'abonnement dans un message d'abonnement

  Délai maximal de réponse = 0 –mis à 0 par l'émetteur et ignoré par les récepteurs– dans un message de rapport ou de réduction.

- Adresse de multicast = adresse du groupe multicast concerné par le message envoyé. Dans les messages de recensement ce champ peut valoir zéro, impliquant ainsi un recensement pour tous les groupes multicast.

Ces messages, dont le format est représenté a la figure 1.27, sont transportés par les paquets IPv6.

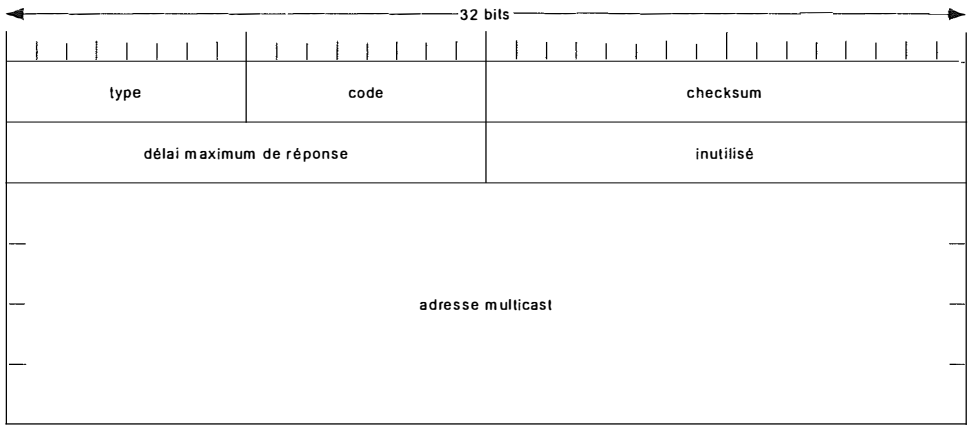



**Fig. 1.27** - Format du message GM

## 1.11 Configuration automatique et contrôle

La configuration automatique des équipements est l'un des attraits principaux de IPv6. Elle porte sur quatre sujets qui sont la découverte des voisins, la configuration des adresses, la renumérotation des routeurs et la découverte du PMTU.

### 1.11.1 Découverte des voisins

Le protocole de découverte des voisins –Neighbor Discovery– permet à un équipement de s'intégrer dans l'environnement local c'est-à-dire le lien sur lequel sont physiquement transmis les paquets IPv6. Il permet un dialogue entre les équipements connectés sur un même support. Il ne s'agit pas à un équipement d'avoir une liste exhaustive de tous les équipements connectés sur le lien, mais uniquement de gérer ceux avec qui il dialogue.

Le protocole de découverte des voisins utilise 5 types de messages ICMPv6. Le champ nombre de sauts de l'en-tête IPv6 contient la valeur 255. Si un équipement reçoit un datagramme avec une valeur inférieure à 255, cela signifie que l'information provient d'un autre réseau et que le datagramme doit être rejeté. Les fonctions réalisées par le protocole sont les suivantes : résolution d'adresses, détection d'inaccessibilité des voisins, configuration, indication de redirection.

Le principe de la résolution d'adresse en IPv6 est très proche du protocole ARP[RFC 826] que l'on trouve dans IPv4. La principale différence vient de l'emploi de messages standards ICMPv6 à la place de la définition d'un autre protocole de niveau 3. Comme pour IPv4, le protocole construit une table de mise en correspondance entre les adresses IPv6 et les adresses physiques.

La fonction qui sert à la détection d'inaccessibilité des voisins NUD –Neighbor Unreachability Detection– n'existe pas en IPv4. Elle permet d'effacer des tables de configuration d'un équipement la liste des voisins qui sont devenus inaccessibles.

Plusieurs fonctionnalités de découverte des voisins sont mises en œuvre :

- Découverte des routeurs. Ce protocole permet aux équipements de déterminer les routeurs qui sont sur leur lien physique. Dans IPv4, ces fonctionnalités sont assurées par le protocole ICMP Router Discovery.
- Découverte des préfixes. L'équipement apprend le ou les préfixes du réseau en fonction des annonces faites par les routeurs. En y ajoutant l'identifiant d'interface de l'équipement, celui construit son ou ses adresses IPv6. Il n'existe pas d'équivalent pour le protocole IPv4 puisque les adresses sont trop courtes pour faire de l'auto-configuration.
- Détection des adresses dupliquées. Les adresses étant configurées automatiquement, il existe un risque d'erreurs en cas de duplication d'un identifiant. Ce protocole teste qu'aucun autre équipement sur le lien ne possède la même adresse IPv6.
- Découverte des paramètres. Ce protocole permet aux équipements d'apprendre les différents paramètres du lien physique. Comme exemple on peut citer la taille du MTU. Il n'existe pas d'équivalent de cette fonctionnalité en IPv4.

L'indication de redirection est utilisée quand un routeur connaît une route meilleure –en nombre de sauts– pour aller à une destination. En IPv4 une indication de redirection ne peut servir qu'à corriger l'adresse du routeur utilisé pour accéder à une machine hors du réseau local. Comme exemple nous pouvons citer le cas d'une machine qui émet ses paquets vers un routeur alors que le destinataire se trouve sur le même segment qu'elle. Dans ce cas le routeur émettra un message de redirction pour que la suite du dialogue se fasse directement.

### 1.11.2 Donnés véhiculées par les messages de découverte des voisins

L'intérêt du protocole de découverte des voisins est d'unifier différents protocoles qui existent dans IPv4. En particulier la plupart des données utilise un format d'options commun –cf. figure 1.28–. Ce qui simplifie la mise en œuvre du protocole. Le message contient trois champs : le type, la longueur en mot de 64 bits et les données.

| | Sollicitation du routeur | Annonce du routeur | Sollicitation d'un voisin | Annonce d'un voisin | Indication de redirection |
|---|---|---|---|---|---|
| Adresse physique de la source | Présent | Présent | Présent | | |
| Adresse physique de la cible | | | | Présent | Présent |
| Information sur le préfixe | | Présent | | | |
| En-tête redirigée | | | | | Présent |
| MTU | | présent | | | |

**Fig. 1.28** - Utilisation des options dans les messages de découvrte des voisins

#### 1.11.2.1 Adresse physique de la source/cible

La figure 1.29 donne le format de ses options. Le type 1 est réservé à l'adresse physique de la source et le type 2 à celle de la cible.

Le champ longueur est la taille en mots de 64 bits de l'option. Dans le cas d'une adresse MAC d'une longueur de 6 octets, il contient la valeur 1.

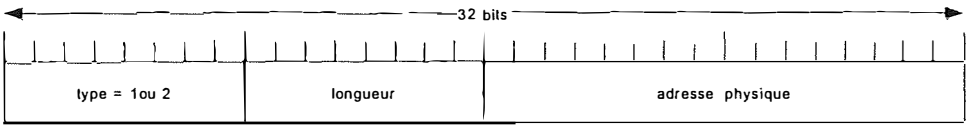



**Fig. 1.29** - Format de l'option adresse physique source/cible

#### 1.11.2.2 Information sur le préfixe

Cette option contient les informations sur le préfixe pour permettre une configuration automatique des équipements. Le format de l'option est donné dans la figure 1.30.

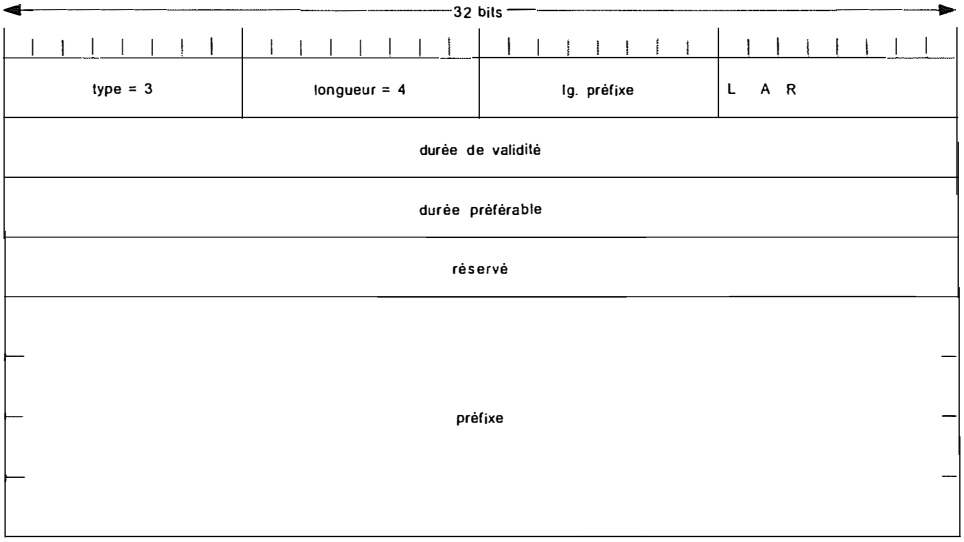



Fig. 1.30 - Format de l'option information sur le préfixe

Le champ *type* vaut 3.

Le champ *longueur* vaut 4.

Le champ *lg. préfixe* indique combien de bits sont significatifs pour le préfixe annoncé dans un champ suivant.

Le bit L indique quand il est à 1 que le préfixe permet d'indiquer que tous les autres équipements partageant le même préfixe sont sur le même lien. L'émetteur peut donc les joindre directement. Dans le cas contraire, le paquet est émis vers le routeur. Si ce dernier sait que l'équipement émetteur peut joindre directement le destinataire, il émettra un message ICMPv6 d'indication de redirection.

Le bit A indique quand il est à 1 que le champ préfixe annoncé peut être utilisé pour construire l'adresse de l'équipement.

Le bit R indique quand il est à 1 que le champ préfixe contient l'adresse globale d'un routeur "agent mère".

Le champ *duré de validité* indique en seconde la durée pendant laquelle le préfixe est valide.

Le champ *durée préférable* indique la durée en secondes pendant laquelle une adresse construite avec le protocole de configuration sans état demeure préférable.

Le champ *réservé* permet d'aligner le préfixe sur une frontière de mot de 64 bits.

Le champ *préfixe* contient la valeur du préfixe annoncée sur le lien. Ce champ a une longueur de 128 bits pour maintenir un alignement sur 64 bits

### 1.11.2.3 En-tête redirigé

Cette option –figure 1.31– est utilisée par le message d'indication de redirection. Les premiers octets du paquet IPv6 qui a provoqué la génération de ce message sont encapsulés comme dans le cas des messages d'erreur ICMPv6.

Le type vaut 4 et la taille de cette option ne doit pas dépasser 1280 octets.

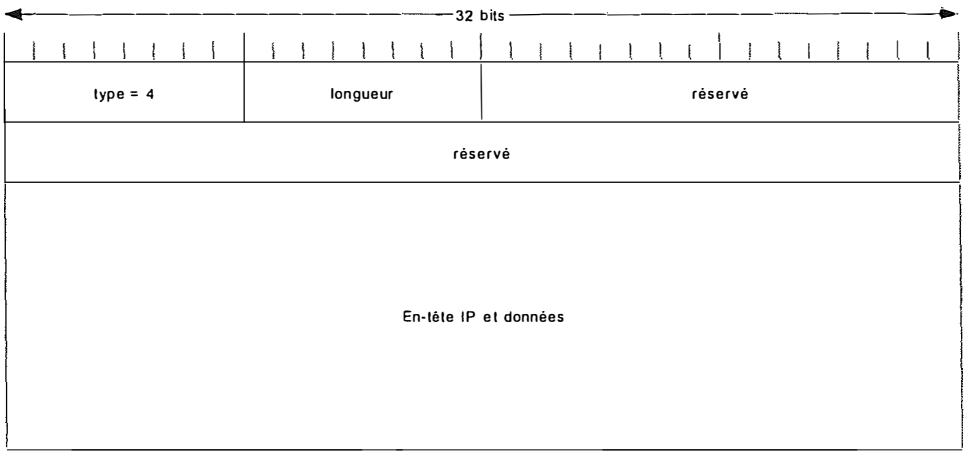



Fig. 1.31 - Format de l'option en-tête redirigée

### 1.11.2.4 MTU –Maximum Transmission Unit–

Cette option permet d'informer les équipements sur la taille maximale des données pouvant être émises sur le lien. Le format de cette option est donné à la figure 1.32. Il n'est pas nécessaire de diffuser cette information si l'équipement utilise la taille maximale permise. Par exemple sur les réseaux Ethernet; les équipements utiliseront la valeur 1500. Par contre pour les réseaux anneau à jeton, il est souvent nécessaire de préciser si les équipements doivent utiliser la valeur maximale permise ou une valeur inférieure. Le champ *type* vaut 5 et le champ *longueur* vaut 1.

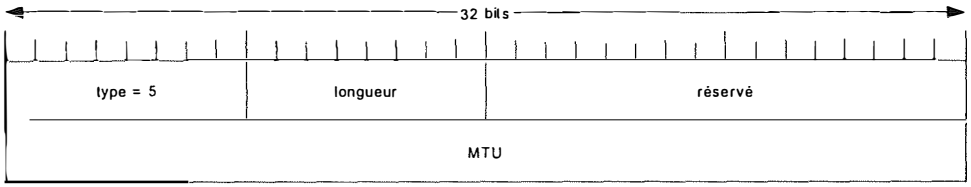



Fig. 1.32 - Format de l'option MTU

### 1.11.3 Messages de découverte des voisins

Les différentes fonctionnalités de découverte des voisins utilisent cinq messages: deux messages pour le dialogue entre un équipement et un routeur, deux pour le dialogue entre voisins et un dernier pour la redirection. Chacun de ces messages peut contenir un ou plusieurs options. Ces dernières, au nombre de quatre, ont été présentées à la section 1.11.2.

### 1.11.3.1 Sollicitation du routeur

Le message de sollicitation d'un routeur –figure 1.33– est émis par un équipement au démarrage pour recevoir des informations du routeur. Ce message est émis à l'adresse IPv6 de multicast réservée aux routeurs sur le même lien FF02::2. Si l'équipement ne connaît pas encore son adresse source, l'adresse non spécifiée est utilisée.

Le champ *option* peut contenir l'adresse physique de l'équipement.

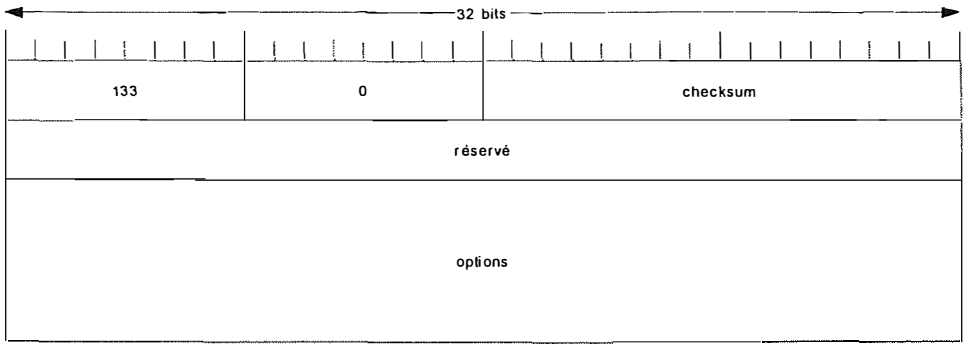



Fig. 1.33 - Format du message *sollicitation du routeur*

### 1.11.3.2 Annonce du routeur

Ce message –figure 1.34– est émis périodiquement par les routeurs ou en réponse à un message de sollicitation d'un routeur émis par un équipement. Le champ adresse source de l'en-tête du datagramme IPv6 contient l'adresse locale au lien du routeur. Le champ destination contient soit l'adresse de l'équipement qui a émis la sollicitation, soit l'adresse de toutes les stations –FF::01–.

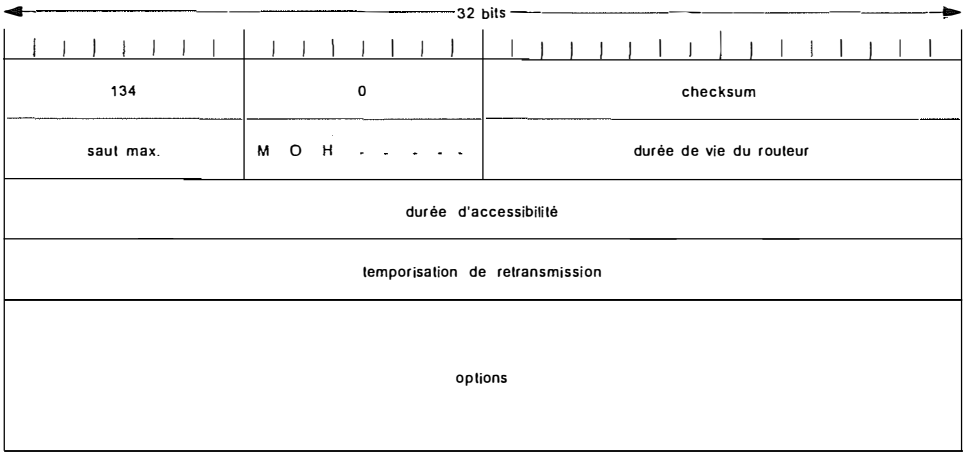



Fig. 1.34 - Format du message *d'annonce du routeur*

Le champ *saut max* non nul donne la valeur qui pourrait être placée dans le champ *nombre de sauts* de l'en-tête IPv6 des paquets émis.

Le bit M indique qu'une adresse de l'équipement doit être obtenue avec un protocole de configuration.

Le bit O sert aussi à la configuration. En fait, si l'adresse ne peut être obtenue d'un serveur, l'équipement procède à une configuration sans état en concatenant au préfixe qu'il connaît son identifiant d'interface.

Le bit H indique que le routeur peut être utilisé comme "agent mère" pour un nœud mobile

Le champ *durée de vie du routeur* donne en secondes la période pendant laquelle l'équipement annonçant effectuera les fonctions de routeur par défaut. La valeur zéro de ce champ indique que l'équipement ne remplit pas les fonctions de routeur par défaut.

Le champ *durée d'accessibilité* indique la durée pendant laquelle une information contenue dans le cache de la machine peut être considérée comme valide. Au bout de cette période, un message de détection d'inaccessibilité est émis pour vérifier la pertinence de l'information.

Le champ *temporisation* de retransmission donne en millisecondes la période entre deux émissions non sollicitées de ce message.

Le champ *options* peut véhiculer les options suivantes : adresse physique de la source, MTU, information sur le préfixe.

### 1.11.3.3 Sollicitation d'un voisin

Ce message –figure 1.35– permet à un équipement d'obtenir des informations d'un voisin – machine située sur le même lien physique–. Dans le cas de la détermination de l'adresse physique, il correspond à la requête ARP du protocole IPv4.

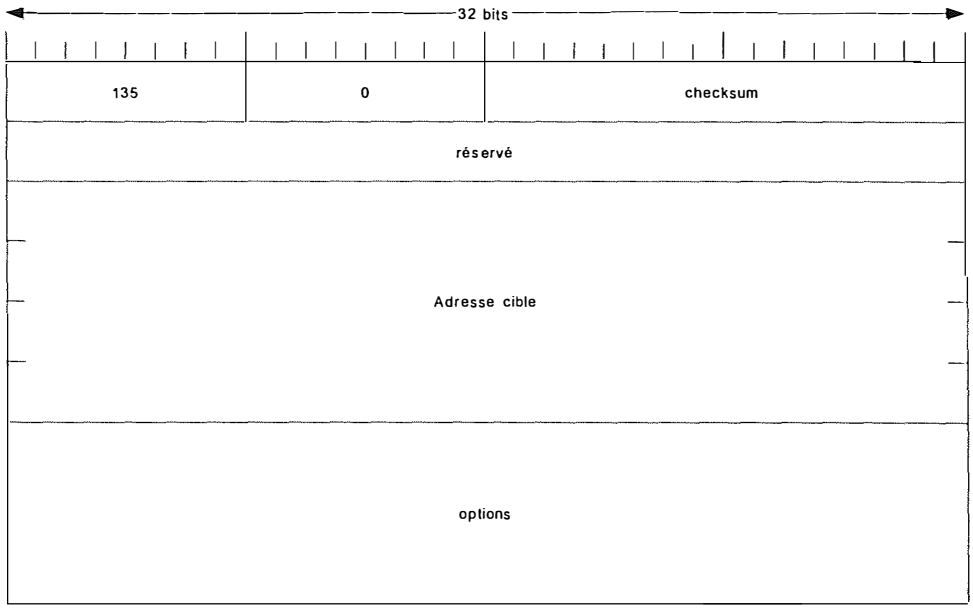



**Fig. 1.35** - Format du message *de sollicitation d'un voisin*

Le champ *adresse cible* contient l'adresse IPv6 de l'équipement.

Le champ *options* contient en général l'adresse physique de la source.

### 1.11.3.4 Annonce d'un voisin

Ce message –figure 1.36– est émis en réponse à une sollicitation. Mais il peut aussi être émis spontanément pour propager plus rapidement une information. Dans le cas de la détermination de l'adresse physique, il correspond à la réponse ARP pour le protocole IPv4.

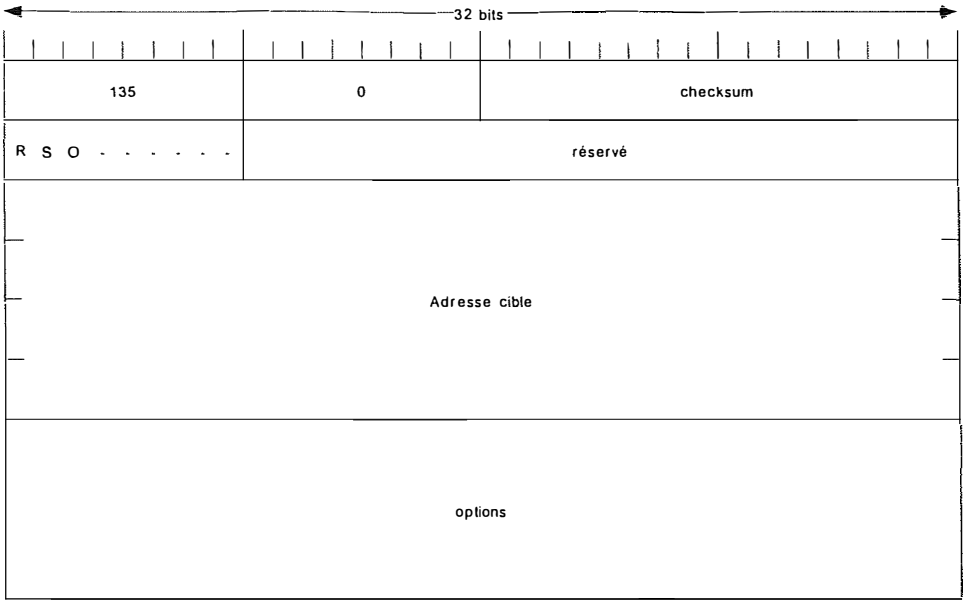



**Fig. 1.36** - Format du message *d'annonce d'un voisin*

Le bit R indique lorsqu'il est à 1 que l'émetteur est un routeur. Ce bit permet de rendre compte du passage d'un équipement de l'état routeur à l'état équipement ordinaire.

Le bit S indique lorsqu'il est à 1 que cette réponse est émise en réponse à une sollicitation.

Le bit O indique lorsqu'il est à 1 que cette annonce doit effacer les informations contenues dans la mémoire cache, en particulier la table contenant les adresses physiques.

Le champ *adresse cible* contient l'adresse IPv6 de l'équipement qui a émis une sollicitation.

Le champ *options* contient l'adresse physique de la source.

### 1.11.3.5 Indication de redirection

En IPv6, la technique de redirection est la même que dans IPv4. Quand un routeur par défaut se rend compte que la route peut être optimisée, celui-ci envoie ce message pour indiquer qu'une route plus courte existe. En fait, avec IPv6, comme le routeur par défaut est appris automatiquement, la route n'est pas nécessairement la meilleure.

Un autre cas d'utilisation particulier à IPv6 concerne des stations situées sur un même lien physique mais ayant des préfixes différents.

Le champ *adresse cible* --cf. figure 1.37– contient l'adresse IPv6 de l'équipement vers lequel les paquets doivent être émis.

Le champ *adresse destination* contient l'adresse IPv6 de l'équipement pour lequel la redirection s'applique. Dans le cas de la redirection vers une machine se trouvant sur le même lien, l'adresse cible et l'adresse destination sont identiques.

Le champ *options* véhicule l'adresse physique du nouveau routeur et l'en-tête du paquet redirigé.

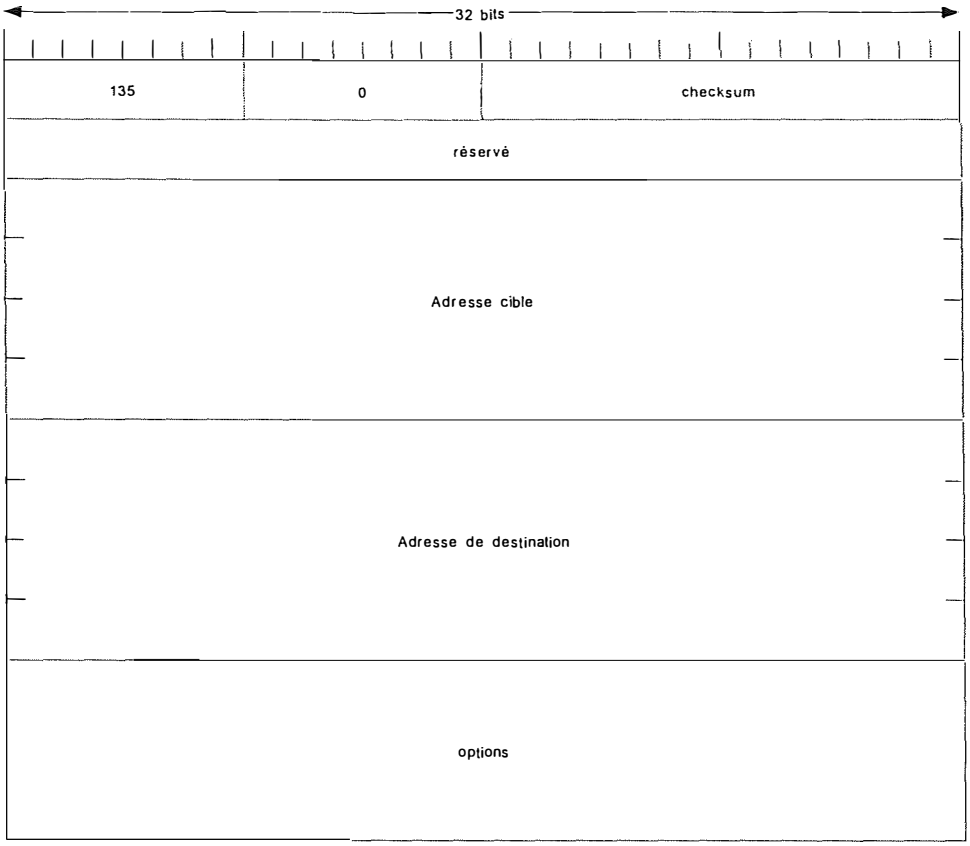



**Fig. 1.37** - Format du message d'indication de redirection

### 1.11.4 Configuration automatique

En IPv4, la configuration d'une interface réseau se fait manuellement. C'est un travail long et fastidieux. Avec IPv6, cette configuration est automatisée ; introduisant de ce fait les caractéristiques de fonctionnement immédiat "plug and play" à l'interface réseau. La configuration automatique signifie qu'une machine obtient toutes les informations nécessaires à sa connexion à un réseau local IP sans aucune intervention humaine.

Outre la construction des tables de routage à partir des messages de redirection discutés à la section précédente, nous allons maintenant étudier un autre aspect de la configuration automatique qui est la configuration d'adresse. Celle-ci a pour objectif l'acquisition d'une adresse quand une machine est attachée pour la première fois à un réseau d'une part, et d'autre part l'obtention de la nouvelle adresse suite à la renumérotation des machines du site après un changement d'opérateur.

L'opération de re-numérotage consiste à changer la partie haute de l'adresse. L'autoconfiguration va servir dans l'attribution du nouveau préfixe.

Le processus d'autoconfiguration d'adresse IPv6 comprend :

- la création d'une adresse lien-local ;
- la vérification de l'unicité de l'adresse créée;
- la détermination de l'adresse unicast globale.

IPv6 spécifie deux méthodes d'autoconfiguration pour l'adresse unicast globale : l'autoconfiguration sans état et l'autoconfiguration avec état.

L'autoconfiguration sans état –stateless autoconfiguration– est utilisée quand la gestion administrative des adresses attribuées n'est pas nécessaire au sein d'un site tandis que l'autoconfiguration avec état -stateful autoconfiguration- est retenue lorsqu'un site demande un contrôle strict dans l'attribution des adresses.

Le rôle du routeur est crucial dans l'autoconfiguration. Il informe la machine par les bits M et O du message d'annonce de routeur sur la méthode à utiliser -avec ou sans état- et fournit éventuellement des messages nécessaires à sa configuration. La valeur de ces bits est donnée par l'administrateur.

L'algorithme d'autoconfiguration d'adresses est le suivant :

- création de l'adresse lien-local;
- vérification de l'unicité de cette adresse sur le lien;
- obtention d'un message d'annonce du routeur. Ce message sert à la détermination de la méthode de configuration pour l'attribution de l'adresse unicast globale;
- S'il y a un routeur sur le lien, la machine applique la méthode indiquée –autoconfiguration avec ou sans état–;
- S'il n'y a pas de routeur sur le lien, la machine essaye d'obtenir l'adresse unicast par la méthode d'autoconfiguration avec état;;
- Si la tentative échoue, alors la machine ne pourra pas communiquer avec des machines autres que celles du lien.

### 1.11.4.1 Création de l'adresse lien-local

A l'initialisation de son interface, la machine construit un identifiant pour celui-ci qui doit être unique au niveau du lien. Dans l'ancienne version du protocole IP, cet identifiant est l'adresse MAC de la machine. Maintenant il est sous le format de l'identifiant EUI-64.

La construction d'une adresse IPv6 consiste à concatener un préfixe avec l'identifiant d'interface. L'adresse lien-local est créée en prenant le préfixe FE80::/64. L'adresse ainsi obtenue est dans un état provisoire non utilisable car son unicité sur le lien est à vérifier. Si l'adresse n'est pas unique alors l'autoconfiguration est arrêtée et une intervention manuelle est nécessaire. Une fois l'unicité sur le lien de l'adresse établie, l'adresse provisoire devient une adresse valide pour l'interface.

### 1.11.4.2 Détection d'adresse dupliquée

Avant d'utiliser une adresse unicast créée, la machine doit exécuter un algorithme de Détection d'Adresse Dupliqué[Hui96] –DAD–. Cet algorithme utilise les messages ICMPv6 "sollicitation d'un voisin" et "annonce d'un voisin". Si une adresse est déjà en service, l'autoconfiguration s'arrête et une intervention humaine devient obligatoire. Une adresse est qualifiée de provisoire pendant l'exécution de l'algorithme DAD et ce jusqu'à la confirmation de son unicité. Une adresse provisoire est allouée à une interface dans le but de lui permettre

de recevoir des messages de sollicitation et d'annonce d'un voisin. Les autres messages reçus sont purement et simplement ignorés. L'algorithme DAD consiste à envoyer un message de sollicitation d'un voisin avec comme adresse cible l'adresse provisoire.

Si au bout d'une seconde –valeur par défaut– aucun message n'est reçu, l'adresse provisoire est considérée comme unique. Elle passe de l'état provisoire à l'état valide et elle est assignée à l'interface.

### 1.11.4.3 Autoconfiguration sans état

L'autoconfiguration sans état ne demande aucune configuration manuelle des machines et une configuration minimum pour les routeurs. Elle se sert du protocole ICMPv6 et peut fonctionner sans routeur. Elle nécessite toutefois un réseau à diffusion.

Le principe de base de l'autoconfiguration sans état est qu'une machine génère son adresse IPv6 à partir d'informations locales et d'informations fournies par un routeur. En fait ce dernier fournit à la machine des informations sur le sous-réseau associé au lien. Il lui donne le la valeur du préfixe en l'occurrence. Ce préfixe provient du message d'annonce du routeur et plus précisément l'option "information sur le préfixe". Comme pour la création de l'adresse lien-local, l'adresse unicast globale est obtenue en concaténant le préfixe avec l'identifiant d'interface. Bien qu'il faille vérifier l'unicité de toutes les adresses unicast, dans le cas d'une adresse obtenue par autoconfiguration sans état cela n'est pas nécessaire dans la mesure où l'identifiant de l'interface a déjà été contrôlé dans la phase de création de l'adresse lien-local. La re-numérotation des machines d'un lien s'effectue au moyen des routeurs qui passent les adresses utilisées dans un état déprécié et annoncent en même temps le nouveau préfixe. Les machines pourront recréer une adresse préférée.

### 1.11.4.4. Autoconfiguration avec état : DHCPv6

Dans l'autoconfiguration avec état, une machine IPv6 obtient les adresses et/ou d'autres informations de configuration par l'intermédiaire d'un serveur. Le mécanisme d'autoconfiguration est bâti sur le modèle client-serveur et repose sur l'utilisation du protocole DHCPv6 –Dynamic Host Configuration Protocol for IPv6–. Il y a un serveur DHCPv6 et la machine est le client IPv6. Le protocole DHCPv6 sert à transmettre les paramètres de configuration du serveur vers le client. Les paramètres transmis doivent permettre à la machine de configurer son interface réseau. Comme informations transmises par le serveur on peut citer: les adresses pour le client, le serveur de nom, le nom de domaine etc. Pour les adresses par exemples, le serveur maintient une table qui lui permet de savoir quelles sont les adresses allouées et celles qui ne le sont pas. L'ensemble des adresses générées par le serveur est créé par l'administrateur du site. Le serveur DHCPv6 ne se trouve pas nécessairement sur le même lien que le client. Dans ce cas, les échanges passent par un relais.

DHCPv6 est un protocole qui se sert du protocole de transport UDP au-dessus de IPv6. Un client envoie des requêtes via le port 547 du serveur et recevra les réponses par le port 546.

Le protocole DHCPv6 se compose de six types de messages qui sont repris en annexe 5.

Le format générique d'un message DHCP est représenté à la figure 1.38.

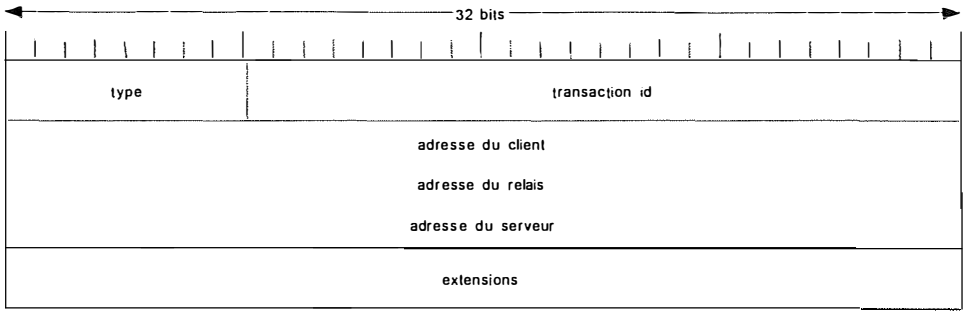



**Fig. 1.38** - Format générique du message DHCP

Le premier octet permet de coder le type de message.

Le *transaction id* est un numéro utilisé pour identifier une requête. Il sera copié par le serveur dans sa réponse.

Les adresses des différents intervenants sont codées sur 16 octets.

Les paramètres de configuration sont codés dans le champ *extensions*.

### 1.11.5 Renumérotation automatique des routeurs

Le protocole de découverte des voisins permet la configuration initiale et la reconfiguration des équipements terminaux de manière automatique. Il ne s'applique pas aux routeurs. Un protocole spécifique basé sur un message ICMPv6 de type 138 permet une reconfiguration automatique des routeurs. Ce message contient un préfixe de test, une opération et des nouveaux préfixes. Lorsqu'un routeur reçoit un tel message, il regarde si le message concerne un des préfixes associés à une des interfaces. Si le préfixe de test est un préfixe de ce préfixe d'interface, alors on applique l'opération contenue dans le message à l'interface. L'opération peut être :

- *Ajouter*. dans ce cas, on ajoute à l'interface les nouveaux préfixes contenus dans le message ;
- *Remplacer*. dans ce cas, on remplace le préfixe sélectionné de l'interface par les nouveaux préfixes contenus dans le message ;
- *Affecter*. dans ce cas, on remplace tous les préfixes associés à l'interface par les nouveaux préfixes contenus dans le message.

Avec un tel protocole, il est possible lors d'un changement de fournisseur d'accès de reconfigurer tous les routeurs du site en quelques messages. Une fois les routeurs reconfigurés, le protocole de configuration automatique permet de reconfigurer tous les équipements du site.

### 1.11.6 Découverte du PMTU

Les datagrammes échangés entre équipements doivent être de taille maximale. Cette taille dépend du chemin suivi et est égale à la plus grande taille autorisée par l'ensemble des liens traversés. Elle est appelée PMTU –Path Maximum Transmission Unit ou unité de transfert de taille maximale sur le chemin–.

Au départ, l'équipement émetteur fait l'hypothèse que le PMTU d'un certain chemin est égal au MTU du lien auquel il est directement attaché. Si la taille des paquets transmis excède la taille maximale autorisée par un lien intermédiaire, alors le routeur associé détruit ces paquets et retourne un message ICMPv6 de type "paquet trop grand" en y indiquant le MTU accepté. A partir des informations reçues du routeur, l'équipement émetteur réduit le PMTU supposé pour ce chemin.

Plusieurs itérations peuvent être nécessaires avant d'obtenir le PMTU permettant à tout paquet d'arriver à l'équipement destinataire sans jamais dépasser le MTU de chaque lien traversé. Le protocole IPv6 garantit que le MTU de tout lien ne peut descendre en dessous de 1280 octets -borne inférieure pour le PMTU-.

Si la détermination du PMTU se fait essentiellement lors des premiers échanges entre les équipements concernés, elle peut également être revue en cours de transfert si suite à un changement de route un lien contraignant est traversé.

L'émetteur vérifie également que le PMTU n'a pas augmenté en envoyant de temps en temps un paquet plus grand. Si celui-ci traverse le réseau sans problème, la valeur du PMTU est augmentée.

# 2 Transition IPv4 vers IPv6

Dans ce chapitre, nous nous intéressons aux techniques à mettre en œuvre pour que les machines équipées du nouveau protocole IPv6 puissent communiquer avec celles équipées de l'ancien protocole IPv4. Nous commençons par une présentation des problèmes techniques que soulève le passage des réseaux IPv4 vers les réseaux IPv6. Nous analysons ensuite les mécanismes et les techniques qu'on pourrait mettre en œuvre pour faire communiquer les équipements des deux types de réseaux. Enfin, nous présentons un résumé comparatif des techniques présentées.

## 2.1 Problématique

Il est bien établi que l'espace d'adressage de IPv4 est limité et que les adresses IP 32-bits ne seront bientôt plus disponibles. Puisque l'architecture du nouveau protocole IPv6/IPng apporte une solution au problème de l'espace d'adressage, du moins pour un certain temps, le besoin de déploiement de cette nouvelle version du protocole IP est réel.

Etant donné l'étendue de l'Internet, la transition du protocole IPv4 vers IPv6 va se faire progressivement moyennant une longue période de cohabitation.

Le passage de IPv4 vers IPv6 se fera-t-il par une intégration –transition lente avec cohabitation des deux systèmes– ou par une migration ? Une chose est sûre, il n'y aura pas un jour J où l'ensemble du réseau devra basculer. La période de transition prendra à tout le moins une décennie. Pendant cette période, il y aura des "bulles" IPv6 dans un monde IPv4. Au fur et à mesure qu'IPv6 prendra de l'ampleur, la tendance s'inversera avec des " bulles" IPv4 subsistant dans un monde IPv6. Lorsque la transition sera pratiquement complète, il faudra également prévoir de faire dialoguer des mondes IPv4 entre eux en passant par IPv6.

Le groupe en charge des recommandations pour la transition vers IPv6 à l'IETF propose plusieurs scénarios de transition et de cohabitation mais n'en impose aucun et laisse choisir à chacun le plus adapté aux circonstances. Deux grands types de problèmes sont à résoudre :

- assurer la communication entre les réseaux IPv4 existants et les réseaux de la nouvelle génération IPv6 comme le présente la figure 2.1;
- assurer la communication entre deux réseaux de la nouvelle génération IPv6 éloignés en passant par un monde IPv4 comme le présente la figure 2.2.

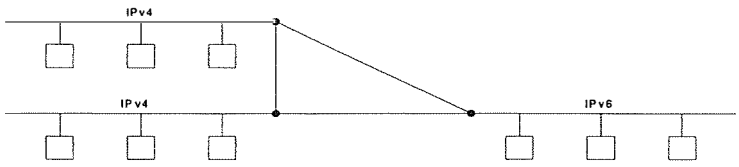



Fig. 2.1 - Réseau IPv6 connecté à IPv4

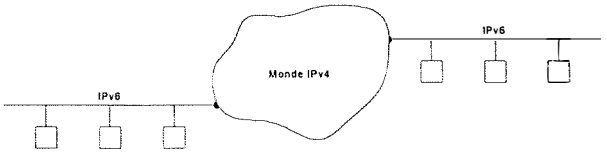



Fig. 2.2 - Réseaux IPv6 connectés par un nuage IPv4

Pour résoudre ces problèmes, la mise à jour des éléments des machines hôtes, des serveurs DNS, des routeurs et des protocoles de routage est d'une importance capitale.

Le serveur DNS apparaît comme le premier composant du réseau qui doit être mis à jour après une allocation des adresses IPv6, celle des autres composants se faisant par les différents outils de transition proposés par l'IETF.

**2.2 Extensions du DNS**

Le DNS –Domain Naming System : système de noms de domaine– a été créé pour regrouper les machines en domaines et leur faire correspondre des adresses IP, ceci afin de permettre une recherche facile d'un équipement dans l'Internet. Au cœur du DNS se trouve un schéma de nommage hiérarchique fondé sur la notion de domaine et une base de données répartie qui implémente ce schéma de nommage. Il est principalement utilisé pour faire correspondre aux adresses IP les noms d'hôtes et les destinations de courrier électronique, mais on peut également l'utiliser en fonction d'autres objectifs.

A chaque domaine, qu'il ne contienne qu'un hôte ou qu'il soit de haut niveau, on associe un ensemble d'enregistrements de ressources.

Lorsque le domaine ne comporte qu'un seul hôte, l'enregistrement de ressource le plus courant est tout simplement l'adresse IP.

Un enregistrement de ressource a le format suivant :

Nom_de_domaine Durée_de_vie Classe Type Valeur

Le champ *Type* indique le type de l'enregistrement. Le type le plus important est le type A – adresse–, qui contient l'adresse IP d'un hôte. A chaque hôte d'Internet doit être attribué une adresse IP afin que les autres machines puissent communiquer avec lui. Les hôtes ayant deux connexions réseau ou plus ont un enregistrement de ressource de type A par connexion à un réseau –et donc par adresse IP–.

La taille des adresses IPv6 d'une part et leur attribution automatique d'autre part rendent l'utilisation du DNS incontournable. Par exemple, alors qu'il est possible de mettre une adresse IPv4 dans un URL, il est probable que l'utilisation directe des adresses IPv6 ne sera pas autorisée dans les URL.

Le serveur de noms devient une pièce cruciale dans le déploiement d'IPv6. C'est le premier composant physique à mettre à jour après qu'une allocation d'adresses IPv6 soit faite et qu'un schéma de "mapping" soit disponible. Les règles de configuration automatique relient l'adresse IPv6 à l'adresse MAC de la machine. De cette façon, si la carte Ethernet de la machine est remplacée, l'adresse IP est aussi modifiée. Il faut donc que l'enregistrement d'un équipement dans le DNS se fasse aussi de manière automatique, sinon la configuration automatique n'a aucun sens. C'est pourquoi le mécanisme de correspondance nom-adresse est fondamental en IPv6. Celui-ci repose comme en IPv4 sur les fichiers /etc/hosts et sur le DNS.

Pour supporter le nouveau schéma d'adressage d'IPv6, trois extensions à apporter au DNS ont été définies selon le RFC 1886:

- enregistrement AAAA ;

- nouveau domaine ip6.int ;
- nouvelles requêtes IPv4 afin que celles-ci renvoient des adresses IPv4 et IPv6.

### 2.2.1 Les enregistrements de ressource AAAA

Le DNS est le mécanisme de base de données réparties qui permet d'associer des informations typées à des noms. De cette façon, la correspondance nom→adresse en IPv4 est réalisée en associant au nom de la machine un ou plusieurs enregistrements de classe IN et de type A. Chaque enregistrement contient une valeur qui est une adresse IPv4.

Pour IPv6, un mécanisme analogue est conservé. Un nouveau type d'enregistrement de classe IN et de type AAAA est défini. Tout comme l'enregistrement qui établit la correspondance entre le nom d'une machine et son adresse IPv4, l'enregistrement AAAA établit la correspondance entre le nom d'une machine et son adresse IPv6. Si une machine possède plusieurs adresses IPv6, elle doit avoir plusieurs enregistrements de type AAAA. Une requête AAAA sur une machine particulière retourne dans ce cas tous les enregistrements AAAA correspondants à cette machine. Toutes les adresses n'ont cependant pas leur place dans le service de nom. N'ont pas d'enregistrements, les adresses lien-local, les adresses IPv4-compatibles et les adresses IPv4-mappées.

Le format textuel d'un enregistrement AAAA tel qu'il apparaît dans le fichier de configuration de la base de données DNS est le même qu'en IPv4, à la seule différence que les adresses sont écrites suivant la syntaxe classique des adresses IPv6. Par exemple, la machine parker.kaynes.be est définie dans la base de données du domaine kaynes.be comme suit :

```
parker      IN      AAAA      4321:0:1:2:3:4:567:89AB
```

Toutefois, en IPv6, une adresse a une durée de vie limitée et l'équipement peut changer d'adresse au fil du temps. En particulier, il est prévu que le préfixe d'un site puisse changer. Dans ce cas, les 64 bits de poids fort de toutes les adresses du site changeront de manière identique. Pour faciliter la propagation d'une telle modification, il est proposé de remplacer les enregistrements type AAAA par un nouvel enregistrement de type A6 qui retourne l'adresse en deux parties, une partie basse donnée par l'enregistrement, et une partie haute qui est une étiquette DNS pointant vers un autre enregistrement de type A6.

```
parker              IN    A6    64    ::3:4:567:89AB      net1.kaynes.be
net1.kaynes.be      IN    A6    48    0:0:0:2::           net.kaynes.be
net.kaynes.be       IN    A6    24    0:1:2::             net.provider.com
net.provider.com    IN    A6    0     4321::
```

Le champ numérique en quatrième colonne indique le nombre de bits de poids fort qui sont fournis par l'enregistrement associé à l'étiquette en sixième position.

L'adresse IP de la machine parker est obtenue par la concaténation.

De cette façon, en cas de renumérotation du site, il suffit de modifier un seul enregistrement, celui de net.kaynes.be.

### 2.2.2 Le domaine ip6.int

La correspondance adresse IP→nom est traitée par le DNS en IPv4 grâce à des enregistrements de classe IN et de type PTR dans un domaine particulier *in-addr.arpa.* De façon analogue, la correspondance entre une adresse IPv6 et le nom de la machine associée est faite par un enregistrement de classe IN et de type PTR dans un domaine de nom spécial *ip6.int.* Une adresse IPv6 est représentée par un nom dans le domaine *ip6.int*. Ce nom est déterminé en prenant la représentation hexadécimale de l'adresse, en l'inversant, en séparant chaque chiffre par un point et en rajoutant le suffixe ip6.int. Par exemple, pour la machine parker.kaynes.be dont une adresse est 4321:0:1:2:3:4:567:89AB, l'enregistrement DNS est :

B.A.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.0.1.2.3.4.ip6.int IN PTR parker.kaynes.be

### 2.2.3 Modification de type de requêtes

Les requêtes qui permettent de rechercher dans la base de données les autres enregistrements de type A et de classe NS, MX et MB doivent être redéfinies afin qu'elles puissent traiter les enregistrements de type A et de type AAAA.

## 2.3 Mécanismes de transition de base

Pour résoudre les deux grands types de problèmes soulevés par la transition de IPv4vers IPv6 à savoir d'une part assurer la communication entre les réseaux IPv4 existants et les réseaux de la nouvelle génération IPv6, d'autre part assurer la communication entre deux réseaux de la nouvelle génération IPv6 éloignés en passant par un monde IPv4, deux mécanismes de base sont proposés: l'installation d'une double piles de protocoles IP et l'utilisation des tunnels.

### 2.3.1 Double piles de protocoles IP

Les nœuds qui implémentent les deux versions du protocole IP comme le présente la figure 2.3 sont capables de communiquer directement avec les nœuds IPv4 et les nœuds IPv6 séparément. Ces nœuds nous les appellerons nœud IPv6/IPv4 par opposition au nœuds IPv6-only et IPv4-only qui implémentent une seule version du protocole IP.

Le résolveur d'adresse doit être capable de traiter à la fois les enregistrements de ressource de type A et ceux de type AAAA. Ces nœuds possèdent une adresse IPv4 et une adresse IPv6. Lorsqu'une requête est envoyée au serveur, trois options différentes doivent être traitées: retourner uniquement l'adresse IPv6, retourner uniquement l'adresse IPv4 ou retourner les deux types d'adresse.

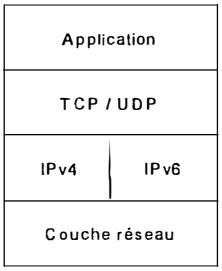



**Fig. 2.3** - Double piles de protocoles IP

### 2.3.2 Utilisation des tunnels

Les mécanismes de tunnels –figure 2.4– décrits dans ce paragraphe permettent une communication entre des réseaux IPv6 isolés dans un monde IPv4.

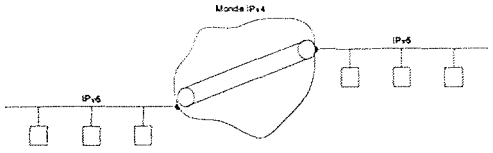



Fig. 2.4 - Schéma d'un tunnel à travers un monde IPv4

#### 2.3.2.1 Mécanismes du tunneling

Le tunneling fournit un moyen d'utiliser une infrastructure IPv4 existante pour transporter un trafic IPv6.

Les machines hôtes et les routeurs IPv6/IPv4 sont capables d'envoyer en mode tunnel les datagrammes IPv6 à travers un monde IPv4 en les encapsulant dans des paquets IPv4.

On note quatre méthodes de tunneling :

- Routeur-à-Routeur;
- Hôte-à-Routeur;
- Hôte-à-Hôte;
- Routeur-à-hôte.

On classe les techniques de tunneling en fonction du mécanisme selon lequel le nœud d'entrée du tunnel détermine l'adresse du nœud de sortie à l'autre bout du tunnel. Dans les deux premières méthodes su-citées, le paquet IPv6 passe par le tunnel dont le nœud de sortie est un routeur. Dans ce genre de tunnel, le routeur qui fait office de nœud de sortie du tunnel décapsule le paquet avant de le relayer vers le destinataire.

Quand l'extrémité finale d'un tunnel est un routeur, le nœud de destination du paquet est différent du nœud final du tunnel. Dans ce cas, les adresses contenues dans le paquet IPv6 ne peuvent fournir l'adresse IPv4 du nœud de sortie du tunnel. L'adresse IPv4 de ce type de nœud doit être déterminée à partir des informations de configuration du nœud d'entrée du tunnel. On utilise le terme de tunnel configuré pour décrire ce type de tunneling où le nœud final du tunnel est configuré explicitement.

Pour ce qui est des deux dernières méthodes de tunneling su-citées, le paquet IPv6 passe par un tunnel sur toute la longueur du trajet parce que le nœud de sortie du tunnel est aussi le nœud de destination. Dans ce cas l'adresse de destination du paquet IPv6 et l'adresse IPv4 de l'en-tête du paquet qui l'encapsule désignent le même nœud. A partir de l'adresse de destination du paquet IPv6, le nœud d'entrée du tunnel peut déterminer automatiquement l'adresse IPv4 de l'extrémité finale du tunnel : est le tunneling automatique. Ce type de tunneling utilise des adresses IPv6 qui contiennent des adresses IPv4 pour permettre une détermination automatique de l'adresse IPv4 de l'extrémité du tunnel.

Les deux techniques de tunneling –automatique et configuré– diffèrent principalement dans la façon dont on détermine l'adresse du nœud de sortie du tunnel, les mécanismes sous-jacents restant les mêmes.

- le nœud d'entrée du tunnel crée un paquet IPv4 encapsulant un paquet IPv6 et transmet celui-ci.
- Le nœud de sortie du tunnel reçoit le paquet IPv4, le ré assemble si nécessaire, enlève l'en-tête IPv4, met à jour l'en-tête IPv6 et traite le paquet.
- Le nœud d'entrée du tunnel doit maintenir certains paramètres de chaque tunnel comme le MTU.

## Encapsulation

L'encapsulation d'un datagramme IPv6 dans un datagramme IPv4 est représentée à la figure 2.5.

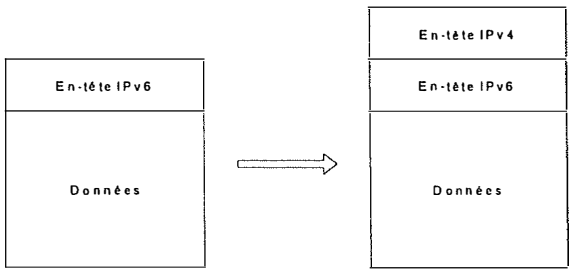



Fig. 2.5- Encapsulation IPv6 dans IPv4

En plus d'ajouter une en-tête IPv4, le nœud d'entrée du tunnel qui fait l'encapsulation doit gérer d'autres paramètres de transmission plus complexes. Celui-ci doit déterminer:

- quand fragmenter un paquet et quand retourner un message d'erreur ICMP à la source lorsque le paquet est trop grand;
- comment reporter les messages ICMPv4 générés par les routeurs dans le tunnel à le source comme message ICMPv6.

## MTU du tunnel et Fragmentation

Le nœud d'entrée du tunnel pourrait voir l'encapsulation comme IPv6 utilisant IPv4 comme couche liaison de données avec un large MTU de 65535-20 octets. Le nœud qui encapsule le paquet pourrait dans ce cas envoyer un message ICMP "Paquet trop grand" à la source lorsque la taille des paquets excède ce MTU. Il est à noter que le chiffre 65535 octets représente la taille maximale d'un datagramme IPv4 et 20 octets est la taille de l'en-tête IPv4.

Ce schéma entraîne une augmentation de la fragmentation avec pour conséquence une diminution des performances et une demande accrue en capacité mémoire pour le réassemblage des fragments.

On peut réduire la fragmentation dans le tunnel au minimum en faisant une découverte du Path MTU –Path MTU Discovery Protocol– dans le tunnel et en stockant le résultat en mémoire. La couche IPv6 du nœud d'entrée du tunnel peut ainsi voir le tunnel comme une couche liaison de données avec un MTU égale au Path MTU de IPv4 moins la taille de l'en-tête IPv4.

Ceci n'élimine pas la fragmentation lorsque le Path MTU de IPv4 entraîne un MTU IPv6 inférieur à 1280 bytes –1280 bytes est le MTU minimum pour IPv6–. Dans ce cas, il faut prendre 1280 comme valeur du MTU. Le nœud d'entrée du tunnel doit utiliser la fragmentation IPv4 pour relayer ces paquets de 1280 bytes.

L'algorithme ci-dessus peut être utilisé par le nœud d'entrée du tunnel pour savoir quand relayer un paquet IPv6 dont la taille dépasse le Path MTU utilisé par IPv4 et quand renvoyer un message ICMP "paquet trop grand".

```
Si ((IPv4 Path MTU - 20) ≤ 1280)
      Si (taille du paquet > 1280) Alors
            Envoyer un message ICMP "paquet trop grand"
            avec MTU = 1280
            Détruire le paquet
      Sinon
            Encapsuler
            Bit DF = 0 (le paquet peut être fragmenté
            par le nœud d'entrée ou par un routeur
            sur le chemin IPv4)
Sinon
      Si (taille du paquet > (IPv4 Path MTU - 20))
            Envoyer un message ICMP "paquet trop grand"
            avec MTU = IPv4 Path MTU - 20
            Détruire le paquet
      Sinon
            Encapsuler le paquet
            Bit DF = 1
```

### Nombre de sauts

Bien que les paquets IPv6 encapsulés traversent plusieurs équipements dans le tunnel, ils sont considérés comme ayant fait un seul saut. C'est ainsi que le champ *nombre de saut* de l'en-tête IPv6 est décrémenté d'une unité lorsqu'il traverse le tunnel. Le tunnel est une boîte noire pour les utilisateurs du réseau. Il n'est pas détecté par les outils tels celui qui permet de tracer la route pris par un paquet.

Par contre,la valeur du champ *TTL* de l'en-tête du paquet IPv4 résultant de l'encapsulation est décrémenté d'une unité à l'intérieur du tunnel chaque fois que le paquet traverse un routeur. Sa valeur initiale dépend de l'implémentation et peut être fixée par l'administrateur du réseau.

### Gestion des messages d'erreur ICMPv4

Les routeurs dans le tunnel envoient des messages d'erreur ICMPv4 au nœud d'entrée lorsqu'une erreur est détectée. C'est le nœud d'entrée du tunnel qui se charge éventuellement d'envoyer un message ICMPv6 au nœud source du paquet.

Le message ICMP "paquet trop grand" est géré comme nous l'avons décrit dans l'algorithme présenté au paragraphe portant sur la fragmentation et le MTU plus haut. La gestion des autres messages d'erreur dépend de la quantité d'informations contenues dans le champ *paquet en erreur*. Ce champ contient le paquet encapsulé qui cause l'erreur. Le nœud d'entrée extrait le paquet IPv6 encapsulé et l'utilise pour envoyer un message ICMPv6 au nœud source.

### Construction de l'en-tête IPv4

Lors de l'encapsulation du paquet IPv6 en datagramme IPv4, les champs de l'en-tête IPv4 sont remplis conformément à l'annexe 6.

### Décapsulation

Quand un routeur ou un hôte IPv6/IPv4 reçoit un datagramme IPv4 qui est adressé à une de ses adresses IPv4 et dont la valeur du champ protocole est 41, il ré assemble le paquet s'il a été fragmenté au niveau IPv4, puis il enlève l'en-tête IPv4 et soumet le paquet résultant à sa couche IPv6.

Les paquets ayant une adresse IPv4 invalide comme adresse multicast, adresse broadcast, 0.0.0.0 et 127.0.0.1 sont ignorés tout comme l'en-tête IPv4.

Après la décapsulation, le nœud doit ignorer les paquets qui ont une adresse IPv6 source invalide. Ceci inclut non seulement les adresses IPv6 de multicast et l'adresse de loopback, mais aussi les adresses sources IPv4-compatibles dont la partie de l'adresse IPv4 est une adresse multicast, une adresse broadcast, 0.0.0.0 ou 127.0.0.1.

Une fois le paquet IPv6 décapsulé, il est traité comme tous les paquets IPv6 à la seule différence qu'il doit être relayé si le nœud a été explicitement configuré pour relayer de tels paquets pour une adresse source IPv4 donnée. Le nœud de sortie du tunnel doit vérifier que l'adresse source du tunnel est une adresse acceptable avant de relayer le paquet décapsulé.

## 2.4 Techniques de transition

### 2.4.1 Connexion des "bulles" IPv6

Les mécanismes de communication par tunnel présentés dans cette partie sont définis dans le RFC2026.

#### 2.4.1.1 Tunnels configurés

Dans le tunneling configuré, l'adresse du nœud de sortie du tunnel est déterminée à partir des informations de configuration du nœud d'entrée. Ce dernier doit en l'occurrence stocker l'adresse du nœud de sortie. Quand un paquet IPv6 est envoyé par un tunnel IPv4, l'adresse du nœud de sortie est utilisée comme adresse destination du paquet IPv4 qui encapsule le paquet IPv6.

La détermination des paquets à envoyer par tunnel est faite à partir des informations de routage du nœud d'entrée et à partir de l'adresse destination.

### Tunnel configuré par défaut

Les machines hôtes IPv6/IPv4 qui sont connectées sur un lien qui ne possède pas un nœud permettant de router les paquets IPv6 doivent utiliser un tunnel configuré pour atteindre un routeur IPv6. Ce tunnel permet à l'hôte de communiquer avec le reste de l'Internet IPv6. Si l'adresse IPv4 d'un routeur IPv6/IPv4 bordant le backbone IPv6 est connue, cette adresse peut être utilisée comme adresse du nœud de sortie du tunnel. Ce tunnel peut être configuré comme route par défaut dans une table de routage IPv6.

### Tunnel configuré par défaut utilisant les adresses anycast IPv4

L'adresse du nœud de sortie d'un tel tunnel par défaut pourrait être l'adresse IPv4 d'un des routeurs IPv6/IPv4 bordant le backbone IPv6. On peut de ce fait utiliser une adresse anycast IPv4 pour atteindre un équipement parmi ces routeurs à périphérie du backbone. Dans cette approche, plusieurs routeurs IPv6/IPv4 au bord du backbone apprennent par un échange de messages l'accessibilité en IPv4 par la même adresse. Tous ces routeurs accepteront les paquets ayant comme adresse destination cette adresse anycast comme leur étant destiné et feront la décapsulation du paquet IPv6. Quand un nœud IPv6/IPv4 envoie un paquet encapsulé à cette adresse, celui-ci sera délivré à un seul routeur bordant le backbone, mais le nœud source ne sait pas lequel. Le routage IPv4 du paquet se fera vers le routeur le plus proche.

L'utilisation d'un tunnel par défaut avec pour adresse du nœud de sortie une adresse anycast garantit une certaine robustesse. En cas de panne d'un des routeurs à la périphérie du backbone, le trafic sera automatiquement routé vers un autre routeur. Des précautions doivent cependant être prises pour éviter que les fragments d'un même paquet n'arrivent chez différents routeurs pour le réassemblage. Ceci peut être fait en évitant de fragmenter les paquets –en assurant un MTU IPv4 d'au moins 1300 octets– ou en évitant de changer régulièrement la route à suivre par les paquets.

#### 2.4.1.2 Tunnels automatiques

Dans un tunnel automatique, l'adresse IPv4 du nœud de sortie est déterminée par l'adresse IPv4-compatible de destination du paquet IPv6 à envoyer par le tunnel. Le tunnel automatique permet au nœud IPv6/IPv4 de communiquer à travers une infrastructure de routage IPv4 sans une configuration préalable des tunnels.

Les adresses IPv4-compatibles sont assignés exclusivement aux nœuds qui supportent le tunneling.

### Configuration de l'adresse IPv4-compatible

Un nœud IPv6/IPv4 avec une adresse IPv4-compatible utilise celle-ci comme une de ses adresses IPv6 pendant que les 32 bits de poids faible de cette adresse font office d'adresse IPv4 de cette interface.

Un nœud IPv6/IPv4 peut acquérir son adresse IPv4-comptatile via les protocoles de configuration d'adresse IPv4. Il peut utiliser n'importe quelle méthode pour acquérir ses adresses IPv4 et les faire correspondre à des adresses IPv4-compatibles en leur ajoutant le préfixe 0.0.0.0.0.0.

Parmi les protocoles qui permettent l'allocation d'une adresse IPv4 à un nœud, on peut citer: DHCP, BOOTP, RARP. On peut même utiliser la configuration manuelle ou autres mécanismes qui permettent d'obtenir une adresse IPv4.

### Opération de tunneling automatique

Dans le tunneling automatique, l'adresse du nœud de sortie du tunnel est déterminé à partir du paquet à transférer. Si l'adresse de destination est une adresse IPv4-compatible alors le paquet peut être envoyé par un tunnel automatique, sinon le paquet ne peut pas être envoyé.

Une implémentation peut avoir une entrée statique dans la table de routage pour le préfixe 0.0.0.0.0.0/96. Tous les paquets qui contiennent ce préfixe dans le champ *adresse destination* sont envoyés au pilote du pseudo-interface qui réalise le tunneling automatique.

Le paquet IPv6 est encapsulé dans un paquet IPv4 selon les règles que nous avons présentées dans la section 2.3.2.1 consacrée aux mécanismes généraux du tunneling. Les adresses source et destination sont déterminées comme suit:

- Adresse destination: les 32 bits de poids faible de l'adresse destination IPv6

- Adresse source: Adresse IPv4 de l'interface via lequel le paquet est envoyé.

Tous les paquets IPv6 dont l'adresse IPv4-compatible de destination contient une adresse IPv4 broadcast, multicast, 0.0.0.0 et 127.0.0.1 sont détruits.

### Utilisation d'un tunnel automatique avec un tunnel configuré par défaut

Le tunneling automatique est souvent utilisé en combinaison avec le tunnel configuré par défaut pour les hôtes IPv6/IPv4 qui n'ont pas de routeur sur leur lien. Ces nœuds sont configurés pour utiliser le tunneling automatique et ont au moins un tunnel configuré par défaut vers un routeur IPv6. Ce dernier est aussi configuré pour faire du tunneling automatique. Les hôtes isolés envoient des paquets vers une adresse IPv4-compatible via un tunnel automatique et les paquets ayant pour destination une adresse IPv6-native sont envoyés via un tunnel configuré par défaut.

### Sélection de l'adresse source

Lorsqu'un nœud IPv6/IPv4 génère un paquet IPv6, il doit sélectionner l'adresse source IPv6 à utiliser. Les nœuds IPv6/IPv4 qui sont configurés pour faire du tunneling automatique doivent avoir aussi bien une adresse IPv4-compatible qu'une adresse IPv6-native. La sélection de l'un ou l'autre type d'adresse déterminera via quelle forme le trafic de retour sera envoyé. Si l'adresse IPv4-comptatible est utilisée comme adresse source, alors le trafic de retour doit être délivré par un tunnel automatique. Si par contre l'adresse source est une adresse IPv6-native alors le trafic retour ne peut être délivré par tunnel automatique. Pour rendre le trafic aussi symétrique que possible, il est recommandé de choisir l'adresse source comme suit:

- Si l'adresse de destination est une adresse IPv4-compatible, alors, il faut utiliser une adresse source IPv4-compatible associée à l'interface assurant le trafic de sortie.

- Si l'adresse destination est une adresse IPv6-native, alors, il faut utliser utiliser l'adresse IPv6 de l'interface de sortie.

- Si un nœud IPv6/IPv4 n'a pas une adresse IPv6-native mais envoie des paquet vers ce type d'adresse, il peut utiliser son adresse IPv4-compatible comme adresse source.

#### 2.4.1.3 6to4

6to4 est une méthode qui permet de connecter des domaines IPv6 via un nuage IPv4. Elle fournit un mécanisme d'assignation d'un préfixe à une adresse IPv6 d'une machine qui a une adresse IPv4 globale. Cette machine peut se connecter avec une autre qui utilise le même mécanisme en envoyant un paquet IPv6 encapsulé dans un paquet IPv4 par les infrastructures IPv4 existantes.

L'objectif de cette méthode est de permettre à des sites –ou hôtes– IPv6 isolés mais attachés à un réseau IPv4 de communiquer entre eux. L'hôte IPv6 qui utilise cette méthode n'a pas besoin d'avoir une adresse IPv4-compatible.

### Allocation du préfixe 6to4 IPv6

Le format du préfixe 6to4 est représenté à la figure 2.6. La valeur du champ *FP* –format préfix– est égale à 0x001 et identifie l'adresse unicast globale du plan d'adressage agrégé. La valeur du champ *TLA* est assignée par IANA pour des besoins du mécanisme 6to4. Elle est de 0x0002. Le préfixe de l'adresse IPv6 est donc 2002::/16.

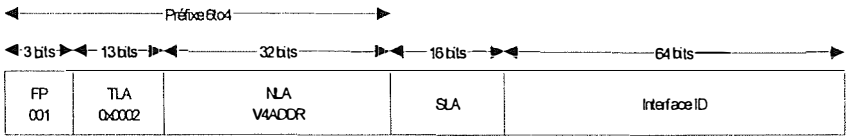



Fig. 2.6 - Format des adresses 6to4

Si un site a au moins une adresse IPv4 V4ADDR qui l'identifie, le préfixe 6to4 est obtenu en concatenant le préfixe 2002::/16 et V4ADDR. L'adresse IPv4 imbriquée dans l'adresse IPv6 6to4 peut être extraite et le paquet IPv6 encapsulé dans un paquet IPv4 peut être délivré par l'infrastructure du nuage IPv4. Ainsi, on n'a pas besoin d'un tunnel configuré pour envoyer des paquets IPv6 entre des sites situés n'importe où sur le réseau IPv4.

### Sélection des adresses

Pour assurer le fonctionnement correct de 6to4 dans des topologies complexes, la sélection des adresses source et destination doit être appropriée.

Si l'hôte source IPv6 a au moins une adresse 2002::/16 et si l'ensemble des adresses retournées par le DNS de l'hôte destination contient au moins une adresse 2002::/16, alors l'hôte source doit faire un choix de l'adresse source et de l'adresse destination à utiliser pour la communication. Le principe de sélection des adresses est le suivant :

- Si un hôte n'a qu'une adresse 6to4 et l'autre a en plus d'une adresse 6to4 une adresse IPv6-native, alors les adresses 6to4 doivent être utilisées pour les deux équipements;

- Si les deux hôtes ont chacune une adresse 6to4 et une adresse IPv6-native, alors les deux équipements peuvent utiliser soient les adresses 6to4, soient les adresses IPv6-native. Le choix doit être configurable avec par défaut l'utilisation des adresses IPv6-native.

### Encapsulation dans IPv4

Les paquets en provenance d'un site 6to4 sont encapsulés dans des paquets IPv4 quand ils quittent le site via sa connexion externe IPv4.

Les paquets IPv6 sont ainsi transmis dans des paquets IPv4. Le champ protocole de l'en-tête IPv4 a la même valeur que dans le cas de l'utilisation des tunnels. L'en-tête IPv4 contient les adresses source et destination du paquet. Une de ces adresses est identique au champ V4ADDR.

La valeur du champ *TTL* est égale à la valeur du champ nombre de sauts de l'en-tête IPv6.

Il faut noter que l'utilisation du mécanisme 6to4 se fait uniquement pendant la période de transition et ne peut être considéré comme une technique permanente. Un site doit migrer de IPv4 à 6to4 et puis vers IPv6.

#### 2.4.1.4 IPv6 over IPv4 –ou 6over4–

La méthode 6over4 définie dans le RFC 2529 permet l'interconnexion des hôtes IPv6 isolés au sein d'un même lien physique sans l'intervention d'un routeur. Les paquets IPv6 sont encapsulés dans des paquets IPv4 sans l'utilisation explicite d'un tunnel. Un lien virtuel utilisant un groupe IPv4 multicast est créé. Les membres du groupe sont limités au lien. Les adresses IPv6 multicast sont associées aux adresses IPv4 multicast afin de permettre une découverte des voisins. Pour un routage entre l'Internet IPv6 et le domaine 6over4, un routeur au moins doit être configuré comme 6over4 sur une interface d'un nœud sur le lien. Les hôtes IPv6 qui utilisent cette méthode n'ont pas besoin d'une adresse IPv4-compatible ou d'un tunnel configuré. Cette méthode est aussi appelée Ethernet virtuel.

### Maximum Transmission Unit

Dans un domaine IPv4, la taille du MTU des paquets IPv6 est de 1480 octets. Cette valeur peut changer par une annonce d'un routeur contenant une option qui spécifie un MTU différent ou par configuration manuelle de chaque nœud.

Si la taille du MTU IPv6 est trop grande, le sous-réseau IPv4 fragmentera le paquet. De ce fait, le bit *DF* de l'en-tête IPv4 ne doit pas être allumé.

### Encapsulation

L'encapsulation du paquet IPv6 dans un paquet IPv6 se fait comme décrit au paragraphe 2.3.2.1 avec le champ protocole de l'en-tête IPv4 valant 41. Une petite valeur doit être allouée au *TTL* afin d'éviter des pertes dans le domaine IPv4. Il est recommandé de prendre une valeur de 8 par défaut.

### Adresses lien-local

L'identifiant d'une interface IPv4 est composé des 32 bits de son adresse IPv4 auxquels on ajoute des zéros pour faire 64 bits au total.

L'adresse lien-local IPv6 associée à une interface IPv4 virtuelle est formée en ajoutant à l'identifiant d'interface construit ci-dessus le préfixe FE80::/64.

### Adresse unicast

L'option adresse source/cible de la couche liaison de données a les caractéristiques suivantes lorsque la couche de liaison est IPv4.

Type : 1 pour adresse source

2 pour adresse cible

Longueur : 1 (en unités de 8 octets)

Adresse IPv4 : adresse 32 bits de l'interface. Cette adresse peut être différente de l'identifiant d'interface obtenue par l'autoconfiguration sans états.

### Adresse multicast

Un paquet IPv6 avec pour adresse destination une adresse multicast DST doit être transmis à l'adresse multicast du lien. Ces adresses multicast doivent être prises dans le bloc 239.192.0.0/16. L'adresse IPv4 multicast obtenue est 239.192.DST[15].DST[16] où DST[i] représente l'octet i de l'adresse destination DST. DST[15] et DST[16] sont en fait les deux derniers octets de l'adresse destination DST.

### Enjeux liés à la transition

Le mécanisme IPv6 over IPv4 fait partie des outils de transition vers la nouvelle génération du protocole IP. Il permet à un site de faire coexister IPv4 avec IPv6.Les deux versions du protocole IP peuvent fonctionner sans avoir à utiliser les adresses IPv4-compatibles ni à configurer les tunnels sur les machines hôtes IPv6. Les interfaces des hôtes et des routeurs doivent être activées en mode 6over4.

Un site IPv4 peut choisir de commencer sa transition vers IPv6 par la configuration d'un routeur afin que celui-ci puisse supporter 6over4 sur une interface connectée sur le domaine IPv4 et une autre interface connectée au réseau Internet IPv6. Tous les hôtes 6over4 du domaine IPv6 peuvent ainsi communiquer avec le routeur et avec l'Internet IPv6 et ceci sans une configuration manuelle de tunnels et sans avoir besoin d'une adresse IPv4-compatible, l'adresse IPv6 des machines hôtes étant allouée par configuration automatique.

Pendant la période de transition, les routeurs doivent annoncer au moins deux préfixes IPv6. Un pour le LAN et un autre pour 6over4. Le préfixe 6over4 doit être unique à l'intérieur du site. 6over4 utilise soit l'adresse site-local soit l'adresse globale.

La gestion par un routeur du LAN et du 6over4 sur la même interface physique entraîne pendant une période de l'autoconfiguration sans état l'utilisation des adresses lien-local avec le même préfixe FE80::/64. Pour éviter la confusion entre la gestion de 6over4 et la gestion du LAN, on doit utiliser les 128 bits de l'adresse lien-local pour distinguer les deux cas.

Avec l'installation d'autres routeurs IPv6 sur le site, les hôtes 6over4 qui deviennent adjacents physiquement aux routeurs peuvent tourner en IPv6 natif. Au niveau des applications IPv6, cela a pour impact une légère augmentation de la taille du MTU.

### 2.4.1. IPv6 Tunnel Broker

L'utilisation de tunnels pour faire communiquer les îlots IPv6 à travers un monde IPv4 sont difficiles à configurer et à maintenir dans les réseaux de taille importante. Cette difficulté est plus importante chez les utilisateurs finaux qui ont déjà une connexion IPv4 et qui veulent entrer dans le monde IPv6.

L'idée de Tunnel Broker, présentée dans le RFC 3053, est une approche alternative des tunnels classiques basée sur l'utilisation de serveurs spéciaux appelés Tunnel Broker qui gèrent automatiquement les demandes de tunnel venant des utilisateurs. Le Tunnel Broker peut être considéré comme un fournisseur d'accès Internet –ISP– IPv6 offrant une connectivité à travers des tunnels 6over4.

La différence fondamentale entre le Tunnel Broker et le mécanisme 6to4 est qu'ils servent différents segments de la communauté IPv6. En effet, le Tunnel Broker est bien adapté aux sites IPv6 isolés, et plus spécialement pour des hôtes isolés dans l'Internet IPv4 qui veulent communiquer avec des réseaux IPv6. L'approche 6to4 a été conçue pour permettre à des sites IPv6 isolés de se connecter sans attendre que l'ISP IPv4 ne fournisse des services IPv6.

### Modèle du Tunnel Broker

Les Tunnels Brokers peuvent être considérés comme des ISPs IPv6 virtuels qui fournissent une connectivité aux utilisateurs déjà connectés à Internet IPv4. Dans l'Internet IPv6 émergeant, on peut s'attendre à ce qu'il y ait plusieurs Tunnels Brokers de telle sorte que l'utilisateur en choisisse un parmi ceux disponibles. La liste des Tunnels Brokers devra être disponible sur un site Web afin de permettre aux utilisateurs de choisir le plus proche et le moins cher.

Le modèle du Tunnel Broker est basé sur un ensemble d'éléments fonctionnels présentés à la figure 2.7

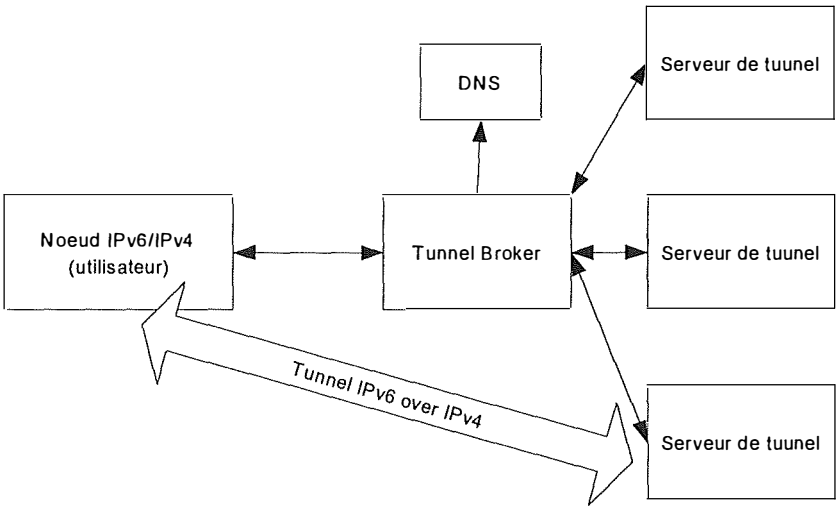



**Fig. 2.7** - Modèle du Tunnel Broker

### Tunnel Broker –TB–

Le nœud utilisateur se connecte au TB, il se fait enregistrer et active les tunnels. Le TB gère la création des tunnels et la destruction des tunnels. Le TB peut gérer plusieurs tunnels à la fois par l'intermédiaire des serveurs de tunnel. Il envoie des ordres de configuration au serveur adéquat lorsqu'un tunnel doit être créé, modifié ou supprimé. Le TB doit également enregistrer l'adresse IPv6 de l'utilisateur et son nom dans le serveur DNS.

Le TB doit avoir une adresse IPv4, une adresse IPv6 n'étant pas obligatoire. La communication entre le Broker et le serveur peut se faire soit en IPv4 soit en IPv6.

### Tunnel Server –TS–

Un TS est un routeur à double piles de protocoles IPv4 et IPv6 connecté à l'Internet global. Bien que recevant des ordres de TB, il crée, modifie et supprime chaque tunnel du côté serveur.

### Utilisation du Tunnel Broker

Le client des services du Tunnel Broker est un nœud IPv6/IPv4 –hôte ou routeur– possédant une liaison Internet IPv4. Pour accéder aux services offert par le TB, le client doit tout d'abord être authentifié. Le TB apparaît ainsi comme un serveur contrôleur d'accès pour les utilisateurs IPv6 interconnectés par IPv4.

Une fois le client authentifié, celui-ci doit fournir les informations suivantes : l'adresse IPv4 côté client du tunnel, le nom qui doit être utilisé pour l'enregistrement des adresses IPv6 attribuées côté client du tunnel dans le serveur DNS, la fonction du client –hôte ou routeur–. Si le client est un routeur IPv6 désirant fournir une connectivité à plusieurs hôtes, il doit fournir le nombre d'adresses IPv6 nécessaires. Ceci permet au TB d'allouer un préfixe IPv6 au client.

Le TB gère les requêtes du client comme suit :

- Désigne le Tunnel Server TS qui doit être utilisé comme extrémité du tunnel.
- Choisit le préfixe IPv6 à allouer au client. La longueur du préfixe est comprise entre 0 et 128. Les valeurs les plus fréquentes étant 48 –préfixe du site–, 64 –préfixe du sous réseau– ou 128 –préfixe de l'hôte– ;
- Fixe la durée de vie du tunnel ;
- Enregistre automatiquement dans le DNS les adresses IPv6 assignées aux extrémités du tunnel.
- Configure le côté serveur du tunnel ;
- Notifie les paramètres de configuration, les noms DNS, les paramètres du tunnel au client

Une fois le configuration terminée, le tunnel IPv6 over IPv4 entre le nœud hôte ou routeur et le TS est prêt à fonctionner. Il permet ainsi à l'utilisateur du TB d'avoir accès aux autres réseaux qui sont connectés au TS.

### Assignation d'adresses IPv6.

Les adresses IPv6 allouées aux deux extrémités du tunnel doivent être des adresses globales appartenant à l'espace d'adressage géré par le TB.

La durée de vie de ces adresses doit être relativement longue et plus longue que la durée de vie des connexions IPv4 des utilisateurs, ce qui permet au client d'avoir une adresse IPv6 semi permanente associées à des noms DNS.

### Gestion du tunnel

Les tunnels actifs sont très consommateurs de ressource sur le TS en terme d'espace mémoire et de temps d'occupation du processeur. Pour cette raison, il est conseillé de maintenir le nombre de tunnels non utilisés le plus bas possible.

Une durée de vie doit être attribuée à chaque tunnel créé. Ce temps pouvant être prolongé par une demande explicite du client.

Pour des utilisateurs qui reçoivent leur adresse IPv4 par un serveur DHCP et qui accèdent à Internet pendant une duré très courte, l'attribution d'une duré de vie au tunnel n'est pas une solution optimale. En effet, l'utilisateur en se reconnectant reçoit une nouvelle adresse IPv4 et doit refaire une nouvelle au Tunnel Broker. Le TB peut forcer la suppression du tunnel pendant un certain temps d'inactivité.

Une autre solution serait d'implémenter un protocole de gestion de tunnel avec une mécanisme qui maintient en vie entre le client et le TB –ou entre le client et le TS– de telle sorte que chaque tunnel peut être libéré dès que l'utilisateur se déconnecte. L'application d'un tel mécanisme demande une mise à jour logicielle de la machine du client afin qu'elle supporte le mécanisme sus-cité.

## 2.4.2 Communication entre nœuds IPv4 et nœuds IPv6

### 2.4.2.1 Modèle de la double piles de protocoles

La solution la plus directe pour faire communiquer les machines IPv4 avec les machines IPv6 est d'inclure une implémentation complète du protocole IPv4 dans les machines hôtes et les routeurs du nouveau système IPv6. Les deux suites de protocoles intégrées constituent ce qu'on convient d'appeler piles de protocoles IP dont le schéma a été présenté à la figure 2.3.

Pendant la transition, nous pourrons avoir trois types de nœuds en fonction de leur capacité à supporter différents protocoles IP :

- nœud IPv4-only : un routeur ou une machine hôte qui implémente uniquement IPv4. Un nœud IPv4-only ne comprend pas IPv6. Ce sont des nœuds qui existaient avant le début de la transition ;
- nœud IPv6/IPv4 : un routeur ou une machine qui implémente IPv6 et IPv4 ;
- nœud IPv6-only : un routeur ou une machine hôte qui implémente uniquement IPv6 et pas de IPv4. Ce type de nœud ne sera utile dans le contexte de l'Internet que lorsque la majorité des systèmes aura basculé vers IPng.

Les nœuds IPv6/IPv4 sont donc capables de transmettre à la fois les paquets IPv4 et les paquets IPv6 et de ce fait, interagir avec tous les systèmes IP du réseau Internet.

La cohabitation de IPv4 et de IPv6 sur un même nœud nécessite une configuration des deux piles, ce qui est complexe. De plus, cela ne solutionne pas le manque d'adresse IPv4 car chaque adresse IPv6 doit correspondre à une adresse IPv4. Si l'on installe la double piles de protocole uniquement sur des serveurs qui servent de conversion, ils seront les seuls à pouvoir dialoguer directement avec le monde extérieur. Par exemple, les postes de travail qui n'auront qu'une pile ne pourront pas être appelées directement par un poste extérieur dans une application de visiophonie. De plus, avec le développement d'Internet à la maison, de nombreux objets communicants devraient apparaître. Ces objets embarqueront des composants qui doivent être simples et peu coûteux. Il devient donc difficile d'intégrer et de configurer des doubles piles dans ces systèmes enfouis. L'Internet à la maison est un des grands domaines qui pousse pour que chaque objet puisse avoir sa propre adresse et ainsi être appelé. Cependant si le décollage des objets connectés se fait avec le protocole IPv4, alors la migration sera très lente car il sera difficile de disposer d'objets ayant une double piles IPv4/IPv6.

### 2.4.2.2 Stateless IP/ICMP Translation Algorithm

L'algorithme SIIT défini dans le RFC 2765 permet de traduire les en-têtes des paquets IPv4 et IPv6 –les en-têtes ICMP incluses– sans recourir aux informations relatives à l'état de la session. Cet algorithme peut être utilisé comme solution permettant à un hôte IPv6 qui ne possède pas une adresse IPv4 permanente de communiquer avec des hôtes qui n'implémentent que IPv4 –ou IPv4-only node–.

Il existe plusieurs cas de figure où les hôtes IPv6-only ont besoin de communiquer avec les hôtes IPv4-only. Ces hôtes IPv6 doivent avoir une implémentation de IPv4 mais sans toutefois que les adresses IPv4 leurs soient allouées. Parmi ces cas de figures nous avons:

- Un nouveau réseau IPv6 avec des équipements qui supportent tous IPv6 et dont les équipements veulent communiquer avec d'autres équipements d'un réseau IPv4 comme représenté à la figure 2.8.
- Un réseau IPv4 existant auquel on ajoute un nombre important d'équipements IPv6. Ces équipements IPv6 doivent également implémenter le protocole IPv4 tel que représenté à la figure 2.9.

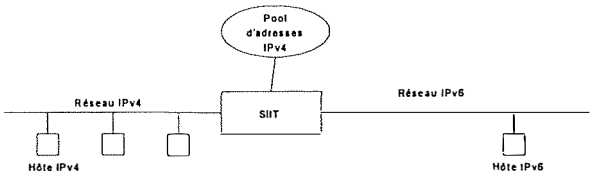



Fig. 2.8 - Utilisation de SIIT pour un réseau IPv6-only

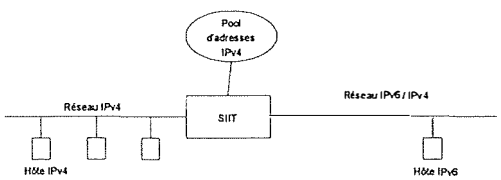



Fig. 2.9 - Utilisation de SIIT pour un hête pour un noueud IPv6-only d'un réseau mélangé IPv6/IPv4

Le protocole SIIT introduit un nouveau type d'adresse appelé adresse IPv4-traduite. Les adresses IPv4 temporaires sont utilisées comme adresses IPv4-traduites et les paquets ainsi émis vont transiter par un traducteur ICMP/IP qui traduit les en-têtes IPv4 et les en-têtes IPv6.

Une adresse IPv4-traduite est de la forme 0::FFFF:0:a.b.c.d où a.b.c.d est une adresse IPv4. Ces adresses permettent de référencer des nœuds qui implémentent le protocole IPv6. Il faut noter que le préfixe 0::FFFF:0:0:0/96 est choisi pour une somme de contrôle nulle afin d'éviter une mise à jour du champ *Checksum* des pseudo en-têtes du protocole de transport correspondant.

### Traduction de IPv4 à IPv6

Lorsqu'un traducteur IPv4-to-IPv6 reçoit un datagramme qui a pour destination un équipement qui n'appartient pas à sa bulle, celui-ci convertit l'en-tête du paquet IPv4 reçu en un en-tête IPv6 et il relaie le paquet à son adresse IPv6 de destination. L'en-tête IPv4 originel est remplacé par un en-tête IPv6, la charge utile du paquet reste inchangée à l'exception des paquets ICMP.

Les détails de la traduction des datagrammes IPv4 en IPv6 sont repris dans l'annexe 7.

### Traduction de IPv6 vers IPv4

Lorsqu'un traducteur IPv6-to-IPv4 reçoit un datagramme dont l'adresse source est une adresse IPv4-mappée, il traduit l'en-tête IPv6 de ce paquet en un en-tête IPv4. L'en-tête IPv6 de départ est supprimé et remplace par un en-tête IPv4. A l'exception des paquets ICMP, l'en-tête de la couche de transport et la partie données restent inchangées.

Il faut cependant tenir compte de la différence entre IPv6 et IPv4 au niveau de la fragmentation et du MTU. Le MTU minimum sur un lien IPv6 est de 1280 bytes. La limite correspondante en IPv4 est de 68 bytes. Il n'est donc pas possible de faire une découverte du MTU lorsqu'un traducteur IPv6-to-IPv4 est sur le chemin de destination. Le nœud IPv6 recevra un message ICMP « paquet trop grand » provenant d'un routeur IPv4. Cependant IPv6 permet une gestion de tels messages en réduisant la taille MTU, mais en deçà de 1280. Quand le MTU est inférieur à la limite 1280, un nœud IPv6 enverra des paquets de 1280 bytes. Ces paquets sont fragmentés par le routeur après leur traduction.

Les détails de la traduction des datagrammes IPv6 en IPv4 sont repris dans l'annexe 7.

#### 2.4.2.3 Network Address Translation - Protocol Translation

NAT-PT définit dans le RFC 2766, permet à un nœud IPv6-only de communiquer avec un nœud IPv4-only. La communication est réalisée par l'utilisation d'un module qui convertit les adresses IPv4 et les adresses IPv6 et garde cet état pendant toute la durée de la session.

Le protocole SIIT traité à la section précédente permet également aux nœuds IPv6 de communiquer avec les nœuds IPv4 mais aucun état de la session n'est requis. SIIT suppose qu'une adresse IPv4 est allouée au nœud IPv6 pour communiquer avec un nœud IPv4.

NAT-PT doit inclure le protocole *Application Layer Gateway –ALG–* pour la traduction des requêtes et des réponses DNS. En effet, une adresse IPv4 est assignée par NAT-PT à un nœud

IPv6 lors de l'initialisation d'une session. Mais l'initialisation d'une session à l'intérieur du lien est différente d'une initialisation hors du lien bien qu'on utilise le même pool d'adresse IPv4. Le protocole DNS-ALG doit pouvoir traduire les requêtes et les réponses de type A en type AAAA et doit pouvoir faire la permutation des adresses IPv6 en adresses IPv4 allouées temporairement.

En combinant le protocole SIIT avec l'allocation dynamique d'adresses du protocole NAT et les ALGs appropriées, NAT-PT fournit une solution qui permet aux applications de communiquer entre les nœuds IPv6-only et IPv4-only.

## Basic-NAT-PT

Nous allons illustrer le fonctionnement du protocole Basic-NAT-PT par un exemple dont le schéma est présenté à la figure 2.10.

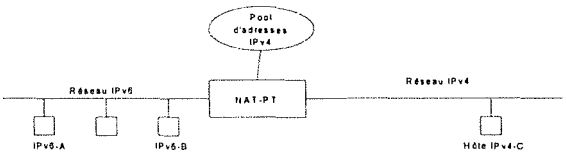



Fig. 2.10 - Schéma d'une communication IPv6 vers IPv4

Soient deux nœuds appartenant à un même lien IPv6 A et B. Ces machines peuvent communiquer avec une machine faisant partie d'un réseau IPv4 C par l'intermédiaire d'un routeur qui implémente le protocole NAT-PT.

Adresse IPv6 du nœud IPv6-A : FEDC:BA98::7654:3210

Adresse IPv6 du nœud IPv6-B : FEDC:BA98::7654:3211

Adresse IPv4 du nœud IPv4-C : 132.146.243.30

NAT-PT possède un ensemble d'adresses attribuables dynamiquement incluant le sous-réseau 120.130.26/24

Supposons que IPv6-A veut communiquer avec IPv4-C. IPv6-A crée un paquet dont les champs adresses sont tels que:

Adresse source SA = FEDC:BA98::7654:3210

Adresse destination DA = PREFIX::132.146.243.30

Le valeur de PREFIX est diffusée par NAT-PT et les paquets avec ce préfixe sont routés vers le routeur NAT-PT.

Le paquet est relayé par NAT-PT après une traduction préalable en IPv4.

Si le paquet en sortie n'est pas un paquet qui initialise une session, alors NAT-PT doit déjà avoir stocké les paramètres de la session; entre autres l'adresse IPv4 assignées à IPv6-A et les paramètres de traduction des en-têtes. Si ce contexte n'existe pas, le paquet est détruit.

Si le paquet initie une session, NAT-PD alloue une adresse IPv4 par exemple 120.130.26.10, et le paquet est traduit en IPv4. Les paramètres de traduction sont stockés en mémoire pendant la durée de la session.

Le paquet IPv4 résultant est tel que : SA = 120.130.26.10 et DA = 132.146.243.30. Les paquets du trafic de retour après l'initialisation de la session sont reconnus comme appartenant à la même session. NAT-PT utilisera les paramètres stockés en mémoire pour faire la traduction des en-têtes. Ces paquets sont tels que SA = PREFIX ::132.146.243.30 et DA = FEDC:BA98::7654:3210.

Il vient de cet exemple que les paramètres à maintenir par NAT-PT sont:

- correspondance entre les adresses IP des deux équipements en communication;
- correspondance entre l'adresse IPv4 allouée et l'adresse IPv6.

### NAPT-PT ou Network Address Port Translation-Protocole Translation

NAPT-PT permet à plusieurs nœuds IPv6 de communiquer avec des nœud IPv4 en utilisant une seule adresse IPv4. Les ports TCP/UDP du nœud IPv6 sont traduits en ports TCP/UDP de l'adresse IPv4 allouée.

Le nombre d'adresses IPv4 qu'on peut allouer aux nœuds IPv6 étant limité, NAPT-PT apporte une solution à cette limite de NAT-PT par l'allocation de numéro de port et non par l'allocation d'une adresse. En effet quand le pool d'adresses IPv4 est épuisé, aucun nœud IPv6 ne peut plus ouvrir une session avec l'extérieur.

Pour illustrer le fonctionnement de NAPT-PT, nous allons partir de l'exemple de la section précédente, à la place d'un routeur NAT-PT nous allons utiliser un routeur NAPT-PT. Toutes les adresses IPv6 peuvent être associées à l'adresse IPv4 120.130.26.10.

Supposons que le nœud IPv6-A veut établir une connexion avec IPv4-C. IPv6-A crée un paquet dont les champs adresse sont tels que:

SA = FEDC:BA98::7654:3210, Port TCP source = 3017

DA = PREFIX::132.146.243.30, Port TCP destination 23

Port de destination : 23

Quand le paquet arrive au module NAPT-PT, celui-ci lui alloue un port TCP à partir de l'adresse IPv4 alloué. Soit :

SA = 120.130.26.10,Port TCP source = 1025

DA = 132.146.243.30, Port TCP destination = 23

Les paquets du trafic de retour en provenance de 132.146.243.30, Port TCP 23 sont reconnus comme appartenant à la même session. NAPT-PT utilisera les paramètres stockés en mémoire pour faire la traduction en IPv6, puis routera les paquets vers le même nœud IPv6-A. Ces paquets sont tels que :

SA = PREFIX ::132.146.243.30, Port TCP = 23

DA = FEDC:BA98::7654:3210, Port TCP 3017

Il vient de cet exemple que les paramètres à maintenir par NAPT-PT sont:

- correspondance entre les adresses IP des deux équipements en communication;

- correspondance entre les numéros de port des équipements en communication;
- correspondance entre l'adresse IPv4 allouée et l'adresse IPv6;
- correspondance entre le numéro de port alloué et le numéro de port de la machine IPv6.

### Utilisation de DNS-ALG pour l'allocation des adresses

Une adresse IPv4 est assignée à un nœud IPv6 quand NAT-PT identifie une initialisation –sur le lien ou en dehors du lien– d'une session. Le début d'une session par un nœud sur le lien est différent de celui d'un nœud en dehors du lien. Toutefois le même pool d'adresse IPv4 est utilisé.

La correspondance entre le nom d'un nœud et son adresse IPv4 est enregistrée dans le DNS par les enregistrements de ressource de type A. Dans le cas des adresses IPv6, l'enregistrement de ressource est de type AAAA.

Un des objectifs de NAT-PT est d'utiliser la traduction seulement au cas où il n'y a pas d'autres moyens de communication entre les nœuds distants.

Assignation d'adresses IPv4 pour les sessions IPv4 vers IPv6

Quand le résolveur du nœud IPv4-C envoie une requête pour IPv6-A, celle-ci est envoyée au DNS du réseau IPv6 et doit traverser le routeur NAT-PT. Le DNS-ALG de NAT-PT va modifier la requête venant de IPv4-C qui est destinée aux enregistrements de ressource de type A comme suit :

- pour les requêtes DNS nom→adresse, changer la requête de type A en type AAAA ou en type A6 ;
- Pour des requêtes adresse→nom, remplacer la chaîne de caractère "in-addr.arpa" par "ip6.int". Remplacer l'adresse IPv4 qui précède la chaîne "in-addr-arpa" avec l'adresse IPv6 correspondante, dans l'ordre inverse.

A l'inverse, lorsque la réponse du DNS du réseau IPv6 traverse le nœud NAT-PT, son DNS-ALG intercepte le paquet et fait les modifications suivantes :

- traduire la réponse DNS de type AAAA ou A6 en type A ;
- remplacer l'adresse IPv6 obtenue par résolution du DNS avec l'adresse IPv4 assignée par NAT-PT.

Pour revenir à l'exemple des paragraphes précédents, IPv4-C essaie de rentrer en communication avec IPv6-A en faisant une requête DNS de type A. La requête va dans le DNS locale et de là elle est propagée au serveur DNS du réseau IPv6. Le DNS-ALG intercepte cette requête et fait la traduction du type A au type AAAA ou A6 et la relaie au serveur DNS du réseau IPv6. Ce dernier répond comme suit :

IPv6-A AAAA FEDC:BA98::7654:3210.

Cette réponse est interceptée et traduite par DNS-ALG par

IPv6-A A 120.130.26.1.

Le DNS-ALG garde cette correspondance entre FEDC:BA98::7654:3210 et 120.130.26.10 dans NAT-PT. L'enregistrement de type A est retourné à IPv4-C.

IPv4-C peut alors initier une session comme suit :

SA = 132.146.243.30, port TCP = 1025

DA= 120.130.26.10 port TCP = 80

La communication peut donc commencer normalement.

Il faut noter que la valeur du champ *TTL* de l'enregistrement de ressource est mise à 0 lorsqu'il passe par le DNS-ALG ceci afin que le serveur ou le client DNS ne stocke cette valeur dans sa mémoire cache.

Assignation d'adresses IPv4 pour les sessions IPv6 vers IPv4

Nous avons vu qu'un nœud IPv6 qui veut communiquer avec un nœud IPv4 doit utiliser un préfixe spécifique PREFIX::/96 devant l'adresse IPv4 du nœud IPv4.

Pour revenir à notre exemple, IPv6-A veut établir une communication avec IPv4-C. Le nœud IPv6-A commence par faire une requête DNS des enregistrements AAAA ou A6 du nœud IPv4-C, le nœud IPv4-C pouvant avoir une adresse IPv4 et une adresse IPv6. Le DNS-ALG de NAT-PT relaie la requête externe AAAA/A6 inchangée et une requête de type A pour IPv4-C. S'il existe un enregistrement de ressource de type AAAA/A6, celui-ci est envoyé au NAT-PT qui le relaie tel quel. S'il y a un enregistrement de type A pour le nœud IPv4-C, le DNS-ALG de NAT-PT traduit cet enregistrement de type A en AAAA/A6.

| | | |
|---|---|---|
| IPv4-C | A | 132.146.243.30 est traduit en |
| IPv4-C | AAAA | PREFIX::132.146.243.30 ou en |
| IPv4-C | A6 | PREFIX::132.146.243.30 |

Le nœud IPv6-A peut ainsi utiliser cette adresse comme toutes les autres adresses IPv6 et le serveur DNS du domaine IPv6 peut stocker cette information dans sa mémoire cache aussi longtemps que le préfixe reste inchangé.

La question qu'on peut se poser est celle de savoir comment le serveur DNS du domaine IPv6 communique avec le domaine IPv4 étant donné qu'il n'y a pas une double piles de protocoles. En fait le DNS externe du domaine IPv4 doit pointer vers une adresse appartenant au pool d'adresses de NAT-PT. Ce dernier maintient une correspondance entre cette adresse IPv4 et l'adresse IPv6 du serveur DNS interne du domaine IPv6. Dans l'autre sens, le serveur DNS du domaine IPv6 pointe vers l'adresse formée par l'adresse IPv4 du DNS externe du domaine IPv4 avec le préfixe PREFIX qui indique que ce n'est pas un nœud IPv6.

### Protocole de traduction

Les détails du protocole de traduction sont donnés par le protocole SIIT. Mais il y a un certain nombre de modifications nécessaires dans la mesure où NAT-PT fait aussi la traduction d'adresses.

Traduction d'en-têtes IPv4 en en-têtes IPv6

Cette traduction est faite comme dans le protocole SIIT à l'exception des champs adresse source et adresse destination.

Adresse source : on ajoute au 96 bits de PREFIX des communications IPv4 l'adresse IPv4 de la source.

Adresse destination : NAT-PT maintenant une correspondance entre l'adresse IPv4 et l'adresse IPv6 du nœud de destination, l'adresse IPv4 de destination est remplacée par l'adresse IPv6 correspondante.

Traduction d'en-têtes IPv6 en en-têtes IPv4

Adresse source : NAT-PT maintient une correspondance entre l'adresse IPv6 source et l'adresse IPv4 allouée à partir du pool d'adresses. L'adresse IPv6 de la source est remplacée par l'adresse IPv4 retenue par cette correspondance.

Adresse de destination : les 32 bits de poids faible de l'adresse de destination sont copiés dans le champ adresse destination de l'en-tête IPv4. En fait les paquets traduits ont une adresse de destination de la forme PREFIX::IPv4/96.

Mise à jour du champ checksum des en-têtes des message ICMP

Le calcul de la somme de contrôle doit tenir compte de l'adresse source et de l'adresse destination obtenues par le protocole PT.

#### 2.4.2.4 Dual Stack Transition Mechanism –DSTM–

L'outil de transition DSTM fait encore partie des sujets de recherche. Le modèle de cette technique de transition est présenté dans des documents de travail de l'IETF. Ces documents de travail sont intitulés *Internet-Drafts*.

### Définitions

Domaine DSTM : zone intranet où les noeuds IPv6 utilisent DSTM pour les communications IPv4.

Serveur DSTM : processus gérant l'ensemble des adresses IPv4 qui seront allouées aux nœuds DSTM.

Client DSTM : processus sur un nœud DSTM qui gère l'adresse IPv4 allouée par le serveur DSTM.

Nœud DSTM : un nœud qui implémente: à la fois IPv4 et IPv6, tunnel 4over6 et est un client DSTM. Un nœud DSTM génère uniquement des paquets IPv6 sur le réseau.

### Modèle DSTM

DSTM est une méthode qui utilise les protocoles existants, c'est-à-dire il ne décrit pas un nouveau protocole. Il définit toutefois des comportements des nœuds, du serveur, du point de sortie du tunnel ainsi que les propriétés du mécanisme d'allocation temporaire d'adresses.

Le mécanisme DSTM a pour objectif de fournir au hôtes IPv6 un moyen d'acquérir une adresse IPv4 pour communiquer avec des hôtes IPv4-only ou communiquer à travers les applications IPv4.

La caractéristique principale de ce mécanisme est que le modèle est transparent pour les applications qui peuvent continuer à fonctionner avec des adresses IPv4. Il est aussi transparent pour le réseau où seuls les paquets IPv6 sont pris en compte. Cet aspect est fondamental pour la transition parce qu'il garantit le fonctionnement des applications malgré la présence d'adresses IPv4 dans la charge utile des paquets.

Les hypothèses suivantes sont faites dans le modèle DSTM dont le schéma est représenté à la figure 2.11:

- un domaine DSTM est contenu dans un intranet et non dans l'Internet;
- les nœuds IPv6 ont une adresse IPv4 temporaire qui leur permet de communiquer avec les nœuds IPv4-only et les applications IPv4;
- l'allocation temporaire d'adresse IPv4 est faite par le serveur DSTM;
- comme une extension d'allocation d'adresse, le serveur DSTM peut aussi fournir un ensemble de numéros de port utilisables par le client. Ceci permet l'utilisation d'une seule adresse IPv4 par plusieurs nœuds DSTM au même moment réduisant ainsi le nombre d'adresses IPv4 nécessaires;
- à l'intérieur du domaine DSTM, les tables de routage IPv4 sont gardées au minimum en faveur du routage IPv6. L'entretien du réseau pour IPv4 est réduit pendant la transition;
- lorsqu'un nœud IPv6 reçoit une adresse IPv4 --et éventuellement un ensemble de ports–, le tunnel 4over6 est utilisé pour transférer les paquets de ce nœud au point de sortie du tunnel où le paquet est décapsulé et relayé en utilisant IPv4. Le serveur DSTM doit pouvoir fournir au client une adresse IPv6 du nœud de sortie du tunnel à utiliser;
- les applications IPv4 existantes ne doivent pas être modifiées pour tourner sur un nœud DSTM;
- un nœud DSTM peut communiquer avec n'importe quel hôte IPv4-only tant que l'adresse destination peut être atteinte à partir du nœud de sortie du tunnel.

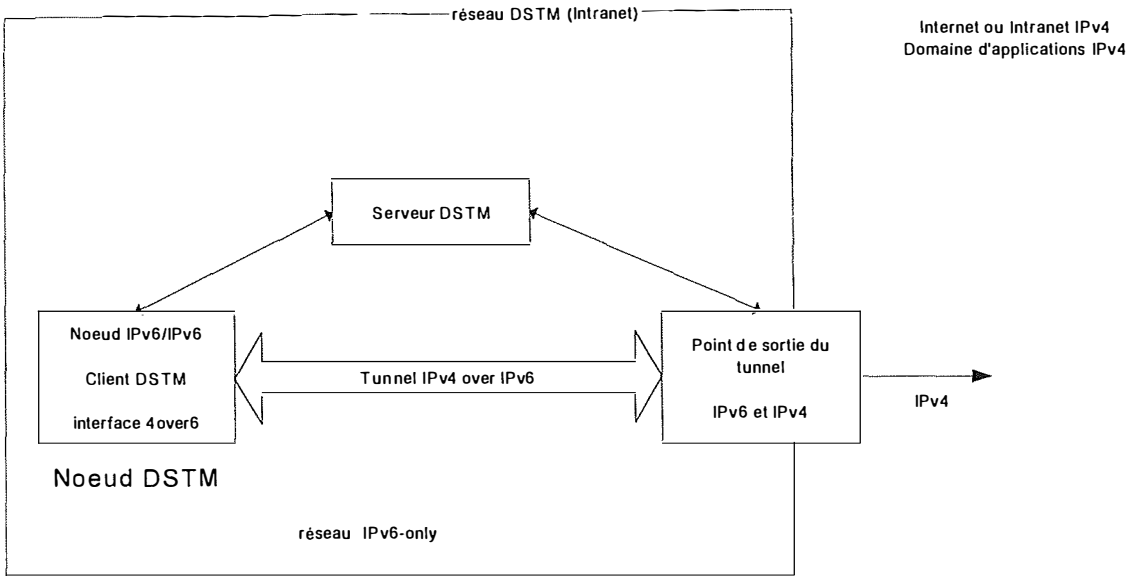



Fig. 2.11-Modèle DSTM

Pour qu'un nœud IPv6 puissent participer au DSTM, il lui faut une double piles de protocoles IP. DSTM n'est pas une solution pour les nœuds IPv6-only.

## Configuration de la pile IPv4

Tant que la communication se fait en IPv6, une adresse IPv4 n'est pas allouée au nœud DSTM. L'hôte peut détecter le besoin d'une adresse IPv4 dans différentes situations: l'adresse destination obtenue du serveur DNS est une adresse IPv4, une application ouvre un socket IPv4, un paquet IPv4 est envoyé au noyau sans qu'il n'y ait une interface prête à le relayer.

Lorsque le premier paquet IPv4 doit être envoyé, le client DSTM doit contacter le serveur DSTM. Le client reçoit du serveur DSTM une adresse IPv4 temporaire ainsi que l'adresse IPv6 du nœud de sortie du tunnel. Si la configuration le permet, le serveur fournit également au client les numéros de ports à utiliser. Ces informations sont utilisées pour la configuration du tunnel 4over6.

## Transfert des paquets IPv4

En l'absence d'une infrastructure de routage IPv4, le nœud DSTM ne peut envoyer des paquets IPv4 dans le réseau. Ceux-ci sont encapasulés dans des paquets IPv6 et envoyés au nœud de sortie du tunnel. Le paquet est décapsulé et relayé au réseau IPv4.

L'encapsulation est faite par l'interface 4over6 du nœud DSTM. Tout le trafic IPv4 peut être envoyé à cette interface par une entrée de la table de routage.

## Traitement des paquets IPv4 par DSTM

Les paquets IPv4 sont envoyés à l'interface 4ovet6. Si cette interface n'est pas configurée, le processus doit être bloqué et le serveur DSTM doit être contacté pour l'allocation d'une adresse IPv4 temporaire. Une fois l'adresse IPv4 allouée, elle est utilisée comme adresse

source de tous les paquets quittant l'interface. Les autres champs du paquet IPv4 sont remplis normalement.

### Construction du paquet IPv6

Quand l'interface 4over6 encapsule un paquet IPv4 dans un paquet IPv6, elle doit déterminer l'adresse IPv6 de destination. En général, ce sera l'adresse du nœud de sortie du tunnel. Cette adresse peut être configurée statiquement ou peut être acquise dynamiquement quand le nœud DSTM obtient l'adresse IPv4 du serveur DSTM.

L'adresse IPv6 du nœud de sortie du tunnel doit être acquise au même moment que le nœud reçoit l'adresse IPv4. Le nœud DSTM peut configurer manuellement l'adresse IPv6 du nœud de sortie du tunnel. Mais ceci ne peut constituer une solution à long terme de la transition.

Lorsque le paquet arrive à l'adresse IPv6 de destination ou au nœud de sortie du tunnel, l'en-tête IPv6 est enlevée et le paquet est traité par la couche IPv4. Le paquet est transféré par le nœud de sortie du tunnel en utilisant l'infrastructure IPv4.

Il y a une association entre les adresses source IPv6 et IPv4.

### Caractéristiques du serveur DSTM

Le serveur DSTM est en charge de l'allocation d'adresses IPv4 temporaires. Celui-ci doit également garantir l'unicité des adresses allouées pendant une duré déterminée. Pour éviter une demande accrue d'adresses IPv4, certaines implémentations pourraient inclure un processus d'allocation de numéros de ports afin de permettre à plusieurs nœud d'utiliser simultanément la même adresse.

Le DSTM doit mémoriser la correspondance entre l'adresse IPv6 du nœud demandant une adresse et l'adresse IPv4 allouée. L'adresse IPv4 est allouée par le serveur pour une durée déterminée et renouvelable.

Le routage dans le monde IPv4 doit assurer que le pool d'adresses IPv4 géré par le serveur DSTM est routé vers au moins un nœud faisant office de point de sortie du tunnel dans le domaine DSTM. En allouant une adresse IPv4 à un nœud, le serveur DSTM doit inclure l'adresse IPv6 du nœud de sortie du tunnel en charge de l'adresse IPv4 allouée.

La communication entre le client DSTM et le serveur DSTM doit se faire en IPv6.

Le Serveur DSTM doit être capable de vérifier l'authenticité d'un client DSTM.

### Applicabilité du modèle DSTM

DSTM est applicable dans un domaine où les hôtes IPv6 souhaitent communiquer avec des hôtes IPv4-only ou à travers des applications IPv4, la communication se faisant soit à l'intérieur d'un intranet soit à travers l'Internet.

DSTM permet aux nœuds implémentant une double piles de protocoles IP de communiquer en utilisant des adresses IPv4 globaux à travers un intranet ou à travers l'Internet où les adresses globales sont nécessaires. Cependant, le mécanisme DSTM peut également être déployé en utilisant des adresses locales afin de permettre aux utilisateurs de solliciter un accès aux services IPv4 de façon temporaire à l'intérieur de l'intranet.

Un mécanisme d'allocation d'adresses est nécessaire dans DSTM. Cette fonctionnalité peut être assuré par DHCPv6.

La configuration dynamique de l'interface 4over6 comme résultat du processus d'allocation d'adresses est la meilleure façon d'exécuter DSTM dans un réseau IPv6. La configuration manuelle du nœud de sortie du tunnel étant déconseillée.

DSTM fait en outre l'hypothèse que le trafic retour en provenance d'un nœud IPv4 vers un nœud DSTM passe par le nœud de sortie du tunnel car celui-ci maintient une correspondance entre les adresses IPv4 et IPv6 des équipements en action.

Les travaux futurs de DSTM pourront permettre au trafic retour en provenance de IPv4 de transiter par plusieurs tunnels et la découverte des nœuds DSTM utilisant les requêtes DNS IPv4.

Nous avons présenté dans ce chapitre différentes techniques qui permettent à des équipements IPv6 de communiquer avec des équipements IPv4. La solution la plus facile serait d'installer une double piles de protocoles IP dans chaque ordinateur connecté à l'Internet. Mais cette solution ne résout pas le problème majeure de IPv4 à savoir le manque d'adresses IPv4 dans la mesure où chaque ordinateur a besoin d'une adresse IPv4 en plus d'une adresse IPv6. Le nombre d'adresses IPv4 étant très limité, on doit allouer dynamiquement des adresses IPv4 aux équipements IPv6 qui veulent communiquer avec des équipements IPv4. Parmi les techniques qui utilisent une allocation dynamique d'adresses IPv4 nous pouvons citer les protocoles NAT-PT, NAPT-PT et DSTM.

Le principe de fonctionnement de NAT-PT est simple. En effet lorsqu'un ordinateur IPv6 veut communiquer avec un ordinateur IPv4, le premier émet un message qui est directement envoyé à NAT-PT qui alloue une adresse IPv4 à l'ordinateur IPv6 et ouvre une session. Dans le cas du protocole DSTM, le mécanisme de communication entre les machines est plus complexe. En effet trois entités physiques différentes interviennent et doivent s'envoyer des messages: demandes d'adresses IPv6, réponse à des requêtes… Tous ces messages contribuent à une augmentation du trafic dans le réseau. Ce qui peut affecter les performances.

Tout comme DSTM, NAT-PT peut être amélioré par une allocation de numéros de port au lieu d'une allocation d'adresses. Mais le problème qui se pose est celui de la stratégie d'allocation de ces numéros de port. En effet, une machine pouvant ouvrir plusieurs sessions, on peut arriver dans une situation où bien qu'une adresse étant allouée à un ordinateur, celui-ci ne puissent ouvrir une autre session parce que les numéros de port correspondants à l'adresse allouée sont tous occupés.

On pourrait imaginer qu'au lieu d'attribuer un numéro de port à la fois on alloue un groupe de numéros de port. Ce qui pourrait réduire le problème; du moins le retarderai.

On pourrait imaginer allouer les adresses par intervalles et à l'intérieur de l'intervalle allouer les numéros de port par le principe de "l'essuie-glace".

Un autre problème à souligner dans le cas de NAT-PT et de DSTM c'est que le trafic entrant et sortant doit passer par le même routeur.

# 3 Analyse des techniques de transition NAT-PT, NAPT-PT et DSTM

L'objectif de ce troisième chapitre est de faire une analyse fonctionnelle et détaillées des techniques de transition NAT-PT, NAPT-PT et DSTM présentées dans le deuxième chapitre de ce travail. Cette analyse s'appuie sur le traitement d'un trafic composé de paquets IP.

Selon le protocole de transport utilisé, un trafic peut contenir différents types de paquets dans une séquence aléatoire. Parmi ces protocoles de transport nous pouvons citer TCP et UDPdont une brève présentation est faite dans la première partie de ce chapitre. La deuxième partie parle de l'analyse fonctionnelle des techniques de transition NAT-PT, NAPT-PT et DSTM. La troisième partie est consacrée à une description technique détaillée de ces technique.

## 3.1 Les protocoles de transport TCP et UDP [Rif98] et [Tan97]

Deux classes de transport des services du protocole IP sont couramment utilisées. Elles correspondent aux deux protocoles UDP –User Datagram Protocol– et TCP–Transmission Control Protocol– de la couche transport du modèle TCP/IP. Les segments transmis par cette couche sont encapsulés, éventuellement après une fragmentation dans des messages IP. Chacun des deux protocoles est identifiable au niveau de ces messages IP par un des champs composant son en-tête.

Les services offerts par ces protocoles UDP et TCP permettent la communication d'informations entre les processus de systèmes distincts. Le multiplexage des informations au niveau de la couche transport repose sur le concept de port. Un service de la couche application sera identifié sur une machine donnée par le protocole de la couche transport dont il utilise les services et par le numéro de port qui l'identifie.

### Le protocole TCP

TCP est un protocole de communication en mode connecté orienté vers la communication de flots d'octets non structurés. Ce protocole offre une meilleure qualité de service mais s'avère plus coûteux. En effet, la demande de connexion est négociée entre les deux interlocuteurs, la qualité de service est assurée par les techniques d'acquittement des messages, de temporisation et de réémission. Un contrôle de flux est réalisé au moyen de fenêtres d'émission et de réception. En outre, il est possible de transmettre des données urgentes – appelées dans la terminologie Internet *Out Of Band Data*–

La figure 3.1 illustre la découpe d'un segment TCP. Chaque segment débute par un en-tête d'une longueur fixe de 20 octets.

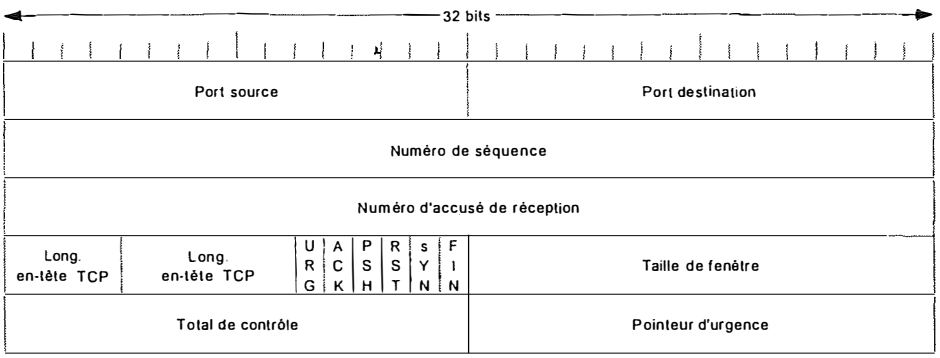



**Fig. 3.1** - Format de l'en-tête TCP [Tan97]

Parmi les champs de l'en-tête TCP, celui composé de six drapeaux d'un bit chacun *URG*, *ACK*, *PSH*, *RST*, *SYN* et *FIN* joue un rôle capital dans la gestion du flux de paquets d'une session.

Le bit *URG* est positionné à 1 si la transmission de données urgentes est en cours.

Le bit *ACK* est positionné à 1 pour désigner la validité de l'accusé de réception.

Le bit *PSH* signifie données poussées. Celles-ci sont remises à la couche application dès leur arrivée.

On utilise *RST* pour réinitialiser une connexion devenue incohérente à cause d'un arrêt brutal d'un ordinateur ou de tout autre accident. Il sert également à rejeter un segment altéré et à refuser une tentative d'ouverture de connexion. En règle générale, si on récupère un segment dont le bit *RST* est à 1, c'est que l'on a un problème.
Le bit *SYN* permet d'établir les connexions. En effet la demande de connexion met ses bits *SYN* à 1 et *ACK* à 0, pour indiquer que le champ accusé de réception en mode superposition n'est pas en cours d'utilisation. Dans la réponse à une demande de connexion comportant un accusé de réception, les bits *SYN* et *ACK* sont donc égaux à 1. Comme on le voit, le bit *SYN* est utilisé pour indiquer à la fois la demande de connexion et communication acceptée, le bit *ACK* faisant la différence entre ces deux possibilités.

On utilise le bit *FIN* pour libérer la connexion. Il indique que l'émetteur n'a plus de données à transmettre. Après avoir libéré la connexion, une application peut toutefois continuer à recevoir des données.

### Le protocole TCP

Avec UDP, les applications peuvent encapsuler des datagrammes IP bruts et les envoyer sans établir de connexion. Dans de nombreuses applications client-serveur, on préfère utiliser UDP plutôt que d'avoir à établir puis à libérer une connexion pour chaque couple de question réponse.

Un segment UDP comporte un en-tête de 8 octets suivi des données. L'en-tête est décrit à la figure 3.2.

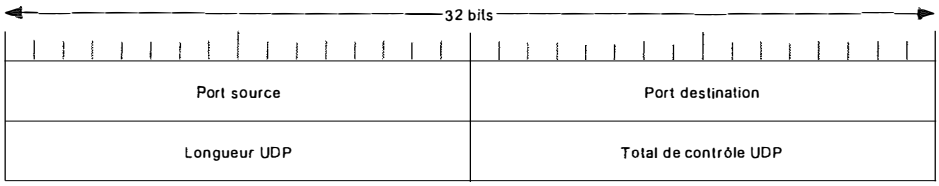



Fig. 3.2 - Format de l'en-tête UDP [Tan97]

## 3.2 Description fonctionnelle

Cette partie traite de l'analyse fonctionnelle des méthodes de transition NAT-PT, NAPT-PT et DSTM en vue d'une évaluation théorique des mécanismes d'implémentation qui peuvent être mis en œuvre.

Cette analyse porte sur le traitements des paquets IP dont le schéma est présenté à la figure 3.3. Elle consiste dans un premier temps à la présentation d'un algorithme de premier niveau qui répond à la question "Comment faire?". Dans un deuxième temps, les différentes fonctionnalités mises en évidence dans la marche à suivre générale sont détaillées.

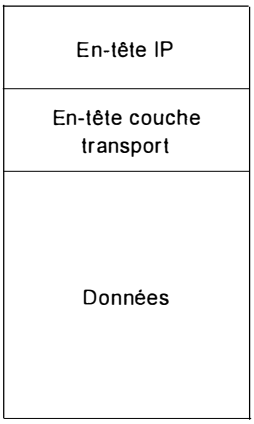



**Fig. 3.3** - Schéma d'un paquet IP

Le format de l'en-tête IP a été présenté au chapitre 1 dans les figures 1.1 et 1.2. Celui de l'en-tête ajouté au niveau de la couche transport correspond au schéma de la figure 3.1 dans le cas du protocole TCP et au schéma de la figure 3.2 dans le cas du protocole UDP.

### 3.2.1 NAT-PT

Le but de la méthode NAT-PT est de faire la traduction d'un flux de paquets IPv6 en un flux de paquets IPv4 et vice versa par une traduction des adresses. Une fois les paquets traduits, ils sont transférés au destinataire. Le routeur NAT-PT sert de passerelle entre le réseau local et l'extérieure. Pour réaliser sa tâche, NAT-PT gère un ensemble d'adresses IPv4. Ces adresses sont allouées aux équipements IPv6 lorsqu'ils désirent communiquer avec les équipements IPv4.

A partir du flux de paquets IPv6 –respectivement IPv4– en entrée, le protocole NAT-PT doit pouvoir générer un flux de paquets IPv4 –respectivement IPv6– sémantiquement équivalent. La traduction de tels paquets passe par l'attribution d'une adresse IPv4 à l'équipement IPv6 durant le temps que durera la session.

Le méthode NAT-PT bien que traduisant des paquets provenant des deux types de réseau suppose que les sessions entre les équipements IPv4 et les équipements IPv6 sont initiées uniquement par ces derniers. Ce qui implique qu'un paquet IPv4 transitant par le routeur NAT-PT dont l'adresse destination n'est pas une adresse allouée du pool IPv4 est détruit.

La méthode générale de traitement des paquets par le processus NAT-PT est présentée dans une marche à suivre.

**Marche à suivre**

```
Pour chaque paquet faire
      Tester l'état de la session en mettant à jour les autres
      Si le paquet appartient au flux d'une session ouverte Alors
            Si paquet de fin de flux Alors
                  Passer en mode "seulement rediffusion et acquittement"
            Traduire le paquet
            Transférer le paquet
            Si paquet de réinitialisation Alors
                  Détruire le paquet
      Sinon (pas de session ouverte)
            Si c'est un paquet IPv4 Alors
                  Détruire le paquet
            Sinon (c'est un paquet IPv6)
                  Si une adresse IPv4 est déjà allouée à l'équipement Alors
                        Ouvrir session
                        Traduire le paquet
                        Transférer le paquet
                  Sinon (l'équipement n'a pas une adresse IPv4)
                        Si une adresse IPv4 est disponible Alors


                   Allouer adresse IPv4 à l'équipement
                              Ouvrir session
                              Traduire le paquet
                              Transférer le paquet
                        Sinon
                              Détruire le paquet
```

Dans la marche à suivre présentée ci-dessus, nous avons eu recours à la description d'actions complexes, dont il nous restera, à ce premier stade de l'analyse, à préciser la teneur et les objectifs précis.

**Ouverture d'une session**

L'ouverture de la session a pour but de garder en mémoire du routeur un certain nombre de paramètres représentant un contexte utile pour une traduction rapide des paquets.

La gestion d'une session est fonction du protocole de transport utilisé. Il faut donc distinguer la gestion d'une session TCP de celle d'UDP.

### *Ouverture d'une session UDP*

UDP est un protocole de transport qui ne demande pas l'établissement d'une connexion entre les deux équipements en communication. NAT-PT ouvre une nouvelle session UDP lorsque le

paquet IPv6 qu'il traite n'appartient pas à une session ouverte. En d'autres termes l'ouverture d'une session a lieu lors du traitement du premier paquet IPv6 qui initie une session. NAT-PT doit garder en mémoire la liste de toutes les correspondances entre adresses IPv6 et adresses IPv4 allouées. La clé d'identification d'une session est composée des éléments suivants :

- adresse IPv6 du premier interlocuteur
- adresse IPv4 du deuxième interlocuteur

Il faut remarquer qu'un équipement IPv6 peut ouvrir des sessions avec plusieurs équipements IPv4.

L'allocation dynamique des adresses entraîne que celles-ci sont attribuées pour une durée déterminée afin qu'un équipement ne monopolise pas l'adresse pendant une durée interminable. De ce fait, pour une session donnée, en plus de sa clé d'identification, les autres paramètres à garder en mémoire sont :

- adresse IPv4 allouée à la machine IPv6
- Durée de vie de l'adresse allouée TTL

### *Ouverture d'une session TCP*

Dans le cas de TCP, le premier paquet envoyé est celui de demande d'une connexion. En plus du fait que ce paquet est le premier envoyé, on peut également savoir à partir du champ drapeau (les bits *SYN* à 1 et *ACK* à 0) qu'il y a demande d'ouverture de session. Dans le contexte qui nous préoccupe, l'ouverture d'une session TCP se fait comme celle d'une session UDP, c'est-à-dire dès qu'il y a un paquet qui n'appartient à aucune session ouverte. Les paramètres de la session à stocker en mémoire restant les mêmes.

L'adresse étant allouée pour une durée limitée, NAT-PT doit à chaque ouverture de session armer un temporisateur qui lorsqu'il expire, la session est fermée et l'adresse désallouée.

**Clôture d'une session**

Elle a pour objectif de supprimer de la mémoire de NAT-PT le contexte créé lors de l'ouverture de la session.

### *Fermeture d'une session UDP*

Comme il n'y a pas d'établissement de connexion, la fin de la session a lieu lorsque le temps alloué à l'adresse s'est écoulé. Mais cette façon de procéder peut entraîner une certaine inactivité pour des sessions dont la durée est inférieure à la durée de vie de l'adresse allouée. Compte tenu de cette constatation, deux solutions sont possibles: soit l'adresse est allouée pour des périodes de temps très courtes; mais dans ce cas des sessions qui durent plus longtemps que le temps alloué seront pénalisées, soit on teste une certaine inactivité de la session par le fait que le temps qui s'écoule entre un paquet et l'attente d'un éventuel paquet suivant est trop long.

### *Fermeture d'une session TCP*

Lorsqu'un équipement envoie le dernier paquet de sa session TCP, il prend le soin de mettre le bit *FIN* à 1 afin de signaler à son interlocuteur qu'il na plus de paquet à envoyer. On ne peut

clôturer la session dès que ce type de paquet apparaît car ce dernier doit être acquitté et le cas échéant il peut y avoir des retransmissions. Nous proposons de faire basculer l'adresse dans un statut spécial appelé état déprécié. Ce basculement du statut de l'adresse est réalisé par la fonctionnalité *Passer au mode "seulement rediffusion et acquittement"*. Ce statut précède toute désallocation d'adresse. Il permet à un équipement de recevoir les acquittements et de faire des rediffusions. Ceci dans les limites de la durée d'allocation de l'adresse.

Lorsque le bit *RST* du drapeau TCP est allumé, il est inutile de se préoccuper des acquittements et des rediffusions car la connexion est devenue incohérente suite à l'occurrence d'un problème. Dans ce cas, a session peut être clôturée sans autre forme de procès.

Allocation et désallocation d'une adresse IPv4 à une machine IPv6

NAT-PT gère un ensemble d'adresses IPv4 qu'il alloue dynamiquement aux machines IPv6 qui veulent communiquer avec des équipements IPv4. Pour qu'il y ait allocation d'une adresse, le pool IPv4 doit avoir au moins une adresse disponible. De ce fait, une adresse IPv4 peut passer par deux statuts : un statut disponible et un statut alloué. L'attribution d'une adresse se caractérise par le basculement du statut disponible au statut alloué. La désallocation étant le basculement inverse.

Une adresse IPv4 est allouée lorsqu'un équipement ouvre sa première session vers un équipement IPv6.

Une machine IPv6 pouvant ouvrir plusieurs sessions avec un ou plusieurs équipements IPv4, une fin de session n'entraîne pas nécessairement une désallocation. Celle-ci n'a lieu que lors de la fermeture de la dernière session ouverte avec un équipement IPv4.

Tester l'état de la session en mettant à jour toutes les autres

Le tableau des sessions ouvertes pourrait être parcouru entièrement par un processus indépendant qui clôture les sessions et désalloue les adresses pour les sessions dont la durée excède la durée de vie de l'adresse IPv4 allouée, pendant que le teste de l'état de la session – ouverte ou non– se fait à partir de la clé composée de l'adresse source et de l'adresse destination par une autre processus. Nous proposons de réaliser ces deux objectifs, à savoir le test de l'état de la fermeture des sessions par la fonctionnalité *Tester l'état de la session en mettant à jour toutes les autres*.

Cette fonctionnalité est capitale pour la gestion de toutes les sessions. En fait elle assure les objectifs suivants:

- Tester si le dernier paquet lu appartient à une session ouverte;
- Tester si l'équipement source possède une adresse IPv4;
- Désallouer les adresses IPv4 lors de la clôture de la dernière session;
- Fermer toutes les sessions qui sont jugées inactives pendant un temps donné.

Le processus reçoit en entrée le paquet qui vient d'être lu. Pour savoir si ce paquet appartient à un flux d'une session ouverte, il suffit de chercher l'élément correspondant à partir de la

valeur de clé composée du couple adresse IPv6 source et adresse IPv4 destination. Pendant cette recherche, le processus réalisera la fermeture des sessions et la désallocation des adresses. La méthode de résolution est la suivante:

```
Pour chaque session ouverte faire
      Si adresse source du paquet = adresse source session Alors
            Adresse IPv4 allouée à l'équipement = oui
      Si paquet lu appartient à la session Alors
            SessionOuverte = oui
      Sinon
            Si TTL dépassé Alors
                  Passer en mode "seulement rediffusion et acquittement"
            Si la session est inactive Alors
                  Fermer la session
                  Si fermeture dernière session d'une machine Alors
                        Désallouer l'adresse IPv4
```

A l'arrivée de chaque paquet, on doit tester si celui-ci fait parti d'une session ouverte. Si c'est le cas alors l'équipement IPv6 dispose d'une adresse IPv4 temporaire.

Le tableau des sessions actives sera parcouru du début jusqu'à la fin pour fermer les sessions inactives, le cas échéant désallouer l'adresses IPv4 s'il s'agit de la fermeture de la dernière session d'un équipement IPv6. Une session est inactive lorsque pendant un temps donné il n'y a pas de flux. Cela nous amène à définir un deuxième temporisateur appelé temporisateur d'inactivité. Lors du traitement d'un paquet d'une session donnée, ce temporisateur est armé à partir de la valeur du timeout d'inactivité. L'expiration de cette valeur entraîne la fermeture de la session.

On ne peut pas compter sur la bonne volonté des utilisateurs pour qu'ils ferment leurs sessions dans les délais. Il faut pouvoir éjecter les machines les plus gourmandes en temps de communication. De ce fait un deuxième temporisateur est armé lors de l'ouverture de chaque session. Celui-ci est initialisé à la valeur de la durée de vie de l'adresse IPv4 allouée à l'équipement IPv6. Si ce temporisateur expire, seuls les acquittements et les rediffusions sont relayés. L'adresse IPv4 étant passée de l'état active à l'état dépréciée.

Destruction du paquet

La destruction d'un paquet intervient lorsque le pool d'adresses IPv4 est vide. Elle consistera à ignorer le paquet et à lire le paquet suivant.

Traduction du paquet

Le traduction des paquets IP utilise le protocole SIIT qui a été présenté au deuxième chapitre et dont les détails sont repris en annexe 2.

Transfert du paquet

Le transfert du paquet suppose que le traitement par NAT-PT est achevé. Donc on lit le paquet suivant.

Comment sait-on qu'une session est devenue inactive?

Une session est devenue inactive à partir du moment où son temporisateur d'inactivité a expiré. Ce temporisateur est armé à chaque passage d'un paquet appartenant à une session donnée. Sa valeur de départ représente le timeout fixé par le gestionnaire du routeur.

Comment sait-on que le temps d'allocation d'une adresse IPv4 est dépassé?

Si le temporisateur correspond à son adresse a expiré. Ce temporisateur est armé lors de l'allocation d'une adresse IPv4 à un équipement IPv6.

Comment sait-on que le paquet qu'on vient de lire appartient à une session ouverte?

Si à partir de la clé constituée par l'adresse source et l'adresse destination on a une entrée dans l'ensemble des sessions ouvertes.

Comment sait-on qu'on est en train de fermer la dernière session d'un équipement?

Le nombre de sessions ouvertes par un équipement est maintenu dans le tableau contenant le pool des adresses IPv4. Lors d'une fermeture de session, un test de ce nombre permet de savoir si on ferme la dernière session d'un équipement.

Comment sait-on qu'une adresse IPv4 est disponible?

Une adresse IPv4 est disponible si son statut est disponible.

Comment sait-on qu'une session est fermée?

Une session fermée ne dispose pas d'entrée dans le tableau des sessions ouvertes contrairement à une session ouverte qui en possède une.

Comment sait-on que le paquet est le dernier d'un flux?

Cette situation n'arrive que dans le cas des sessions TCP où un drapeau indique la fin du flux.

Comment sait-on que nous avons un paquet de réinitialisation?

Lorsqu'une session est devenue incohérente, le bit *RST* du drapeau de l'en-tête TCP est allumé.

*3.2.2 NAPT-PT*

L'amélioration qu'apporte NAPT-PT par rapport à NAT-PT réside dans le fait qu'au lieu d'attribuer des adresses à des équipements IPv6 qui veulent communiquer avec des équipements IPv4, on leur attribut plutôt des numéros de port au sein d'une adresse IPv4. De ce fait une adresse IPv4 peut être attribuée à plusieurs équipements au même moment. L'initialisation d'une communication entre une machine IPv6 et une machine IPv4 se traduit par l'allocation d'une adresse IPv4 et d'un numéro de port à l'équipement IPv6. La marche à suivre ressemble à celle du protocole NAT-PT.

Marche à suivre

```
Pour chaque paquet faire
      Tester l'état de la session et vérifier les autres
      Si le paquet appartient à un flux d'une session ouverte Alors
            Si paquet de fin de session Alors
                  Passer en mode "seulement rediffusion et acquittement"
            Traduire le paquet
            Transférer le paquet
            Si paquet de réinitialisation Alors
                  Détruire le paquet
      Sinon (pas de session ouverte)
            Si c'est un paquet IPv4 Alors
```

```
            Détruire le paquet
      Sinon (c'est un paquet IPv6)
            Si l'équipement a déjà une adresse IPv4 Alors
                  Si numéro de port libre au sein de l'adresse Alors
                        Allouer numéro de port
                        Ouvrir une session
                        Traduire le paquet
                        Transférer le paquet
                  Sinon (pas de port libre)
                        Détruire le paquet
            Sinon (l'équipement n'a pas une adresse IPv4)
                  Si adresse et numéro de port disponible Alors
                        Allouer adresse et numéro de port
                        Ouvrir une session
                        Traduire le paquet
                        Transférer le paquet
                  Sinon
                        Détruire le paquet
```

Dans le cas de NAPT-PT, la clé d'identification d'une session est composée des éléments suivants:

- adresse IPv6 du premier interlocuteur
- numéro de port du premier interlocuteur
- adresse IPv4 du deuxième interlocuteur
- numéro de port du deuxième interlocuteur

En plus de la clé d'identification, les autres paramètres que le routeur doit garder en mémoire sont :

- adresse IPv4 allouée à l'équipement IPv6
- Numéro de port alloué
- Durée d'allocation du numéro de port

Tester l'état de la session et vérifier les autres

Comme dans le cas de NAT-PT, cette fonctionnalité assure la gestion de toutes les sessions ouvertes. Elle consiste à:

- tester si le dernier paquet lu appartient à une session ouverte
- Désallouer les numéros de port lors de la clôture de la dernière session
- Fermer toutes les sessions qui sont jugées inactives pendant un temps donné.

Pour savoir si un paquet appartient à un flux d'une session ouverte, il suffit de chercher l'élément correspondant à partir de la valeur de la clé. Pendant cette recherche, le processus réalisera la fermeture des sessions et la désallocation des numéros de port. La marche à suivre est la suivante:

```
Pour chaque session ouverte faire
      Si paquet courant appartient à la session Alors
            SessionOuverte = oui
      Sinon
            Si la durée de vie du port est dépassée Alors
                  Passer en mode "seulement rediffusion et acquittement"
```

```
Si la session est inactive Alors
      Fermer la session et restituer le numéro de port
      Si fermeture dernière session pour d'une machine Alors
            Désallouer l'adresse IPv4
```

Passer en mode "seulement rediffusion et acquittement"

Cette fonctionnalité a pour objectif, comme dans le cas de la désallocation des adresse IPv4 par NAT-PT, de faire basculer le statut du numéro de port de *préféré* à *déprécié*.

Allocation et désallocation de numéros de port

Un équipement peut maintenir une ou plusieurs sessions avec un ou plusieurs équipements. Dans ce cas, ce qui change c'est le numéro de port, l'adresse IP restant la même. Donc à chaque adresse IP correspondent plusieurs numéros de port dont les allocations et les désallocations se font comme de manière identique à celles des adresses IPv4. Chaque fois qu'on veut ouvrir une session, un numéro de port est attribué si l'équipement IPv6 dispose déjà d'une adresse IPv4. Dans le cas contraire, outre l'allocation d'un numéro de port, celle d'une adresse s'impose.

Allocation et désallocation d'adresses IPv4

L'allocation d'une adresse IPv4 à un équipement IPv6 a lieu lorsque ce dernier veut communiquer avec un équipement IPv4. Elle est suivie d'une allocation d'un numéro de port de communication. Pour ne pas attribuer une deuxième adresse IPv4 à un équipement lors de la réception d'un paquet qui n'appartient pas à une session ouverte, il faut s'assurer que l'équipement ne dispose pas déjà d'une adresse. Dans ce cas seule un numéro de port est attribué.

Comment sait-on qu'un équipement a déjà une adresse IPv4?

Un équipement à qui une adresse IPv4 a déjà été allouée peut ouvrir une autre session avec le même ou un autre équipement. Dans ce cas, NAPT-PT doit tout simplement lui allouer un numéro de port de communication, l'adresse IPv4 allouée restant inchangée. D'où la nécessité de vérifier qu'un équipement possède déjà une adresse IPv4. Une adresse est allouée à un équipement si parmi les éléments du tableau des sessions ouvertes, son adresse y figure parmi les adresses IPv6

Comment sait-on qu'un numéro de port est libre au sein d'une adresse?

Si étant donnée une adresse IPv4, on peut trouver au moins un numéro de port libre.

Comment sait-on qu'une adresse et un numéro de port sont disponibles?

La disponibilité d'un numéro de port entraîne la disponibilité d'une adresse.

### *3.2.3 DSTM*

Le mécanisme DSTM fait intervenir trois entités dans le traitement des paquets en provenance des équipements IPv6 qui veulent communiquer avec ceux qui implémentent IPv4. Ces trois entités sont: le serveur DSTM, le client DSTM et le TEP comme nous l'avons présenté à la figure 2.11du deuxième chapitre.

Marche à suivre

Pour chaque paquet faire

```
Si l'équipement IPv6 possède une adresse IPv4 Alors
     Encapsuler le paquet
     Envoyer le paquet au TEP
     Décapsuler le paquet
     Tranférer le paquet
Sinon
     Envoyer une requête au serveur DSTM
     Traiter la requête du client DSTM
     Envoyer la réponse au client DSTM
     Si une adresse IPv4 a été allouée Alors
          Envoyer la correspondance des adresses au TEP
          Configurer la couche IPv4
          Encapsuler le paquet
          Envoyer le paquet au TEP
          Décapsuler le paquet
          Transférer le paquet
     Sinon
          Détruire le paquet
Mettre à jour le tableau des correspondances IPv4-IPv6
```

Les trois processus sus-cités communiquent entre eux en s'envoyant des message par le réseau.

Mettre à jour le tableau des correspondances IPv4-IPv6

La mise à jour du tableau des correspondances consiste à la vérification des sessions inactives et des sessions dont la durée de communication dépasse la durée de vie de l'adresse IPv4 allouée. Un premier temporisateur est armé lors de l'allocation d'une adresse IPv4 au client. Ce temporisateur expire lorsque la durée de vie de l'adresse est écoulée. Le deuxième temporisateur est celui d'inactivité, il est armé chaque fois qu'un paquet d'une session donnée passe par le TEP.

Traiter la requête du client DSTM

La requête du client DSTM consiste en une demande d'adresse IPv4. Le serveur DSTM consulte son pool et s'il y a une adresse disponible, celle-ci est allouée au client pour une durée limitée. Le serveur garde en mémoire la correspondance entre les adresses IPv6 et les adresses IPv4 correspondantes ainsi que la durée de vie de ces adresses.

Désallocation des adresses IPv4

La désallocation de l'adresse IPv4 attribuée à une machine IPv6 intervient soit parce que le temps d'allocation est dépassé; dans ce cas une notification est faite au client afin que celui-ci fasse une autre demande explicite. Mais pour éviter le mécanisme d'allocation et de désalocation, nous allons désallouer les adresses lorsqu'un équipement devient inactif pendant un temps donné. L'équipement qui est à même de détecter une quelconque inactivité d'un client DSTM est le TEP par lequel passent tous les paquets faisant partie d'un même flux.

Renvoyer la réponse au client DSTM

Deux cas peuvent se produire au cours du traitement de la requête du client par le serveur. S'il n'y a pas une adresse disponible, alors une réponse négative est notifiée au client. Si au contraire une adresse est disponible alors celle-ci est envoyée au client avec la durée pendant laquelle l'équipement peut l'utiliser. Pour éviter une surcharge du réseau, on pourra imaginer

que le serveur ne réponde que dans le cas où une adresse est disponible. Dans ce cas le client peut renvoyer une requête après un temps donné.

Envoyer la correspondance des adresses au TEP

Afin de pouvoir encapsuler les paquets du flux entrant dans le domaine DSTM en provenance du monde IPv4, le TEP doit avoir une copie des correspondances adresse IPv6 et des adresses IPv4. Le TEP peut également constituer dynamiquement ces correspondances lors de la décapsulation des paquets du flux sortant du domaine DSTM. Nous allons supposer que le TEP et le serveur DSTM ont accès à la même copie des données.

Encapsuler le paquet

Le paquet IPv4 est encapsulé dans un paquet IPv6 conformément à la technique 6over4 présentée au paragraphe 2.4.1.4 du chapitre 2.

Transférer le paquet

Le paquet décapsulé par le TEP est envoyé vers le destinataire

Décapsuler le paquet

Le paquet IPv6 est décapsulée conformément à la technique 6over4 présentée au paragraphe 2.4.1.4 du chapitre 2.

Envoyer le paquet au TEP

Le paquet IPv4 encapsulé dans un paquet IPv6 est envoyé au TEP

Configurer la couche IPv4

Configurer l'interface IPv4 et le tunnel

Détruire le paquet

On passe au traitement du paquet suivant

## 3.3 Description technique détaillée

L'analyse fonctionnelle faite, il nous faut préciser complètement ce que signifie et recouvre chacune des actions complexes envisager, pour qu'il ne puisse subsister aucune ambiguïté et que le problème soit parfaitement défini. Autrement dit, nous allons définir le "quoi faire?". Pour cela, nous allons dans un deuxième niveau traduire les marches à suivre générales de la section 3.2 dans des diagrammes états-transitions. L'avantage de ces diagrammes réside dans leur facilité de compréhension de l'analyse du problème et le découpage de la solution en de petites actions que l'on peut facilement spécifier.

### 3.3.1 NAT-PT

Ensemble des adresses IPv4

L'ensemble des adresses IPv4 baptisé Pool est une liste dont la clé d'accès aux différentes valeurs est l'attribut qui représente l'adresse IPv4.

| **Pool** | | | |
|---|---|---|---|
| **Attribut** | **Format** | **Valeur par défaut** | **Description** |
| Adr | Format : x.x.x.x | | Adresse IPv4 |
| Adr_statut | Entier<br>0 :disponible<br>1 :préférée<br>2 : dépréciée | 0 | Statut de l'adresse IPv4 |

| | | | |
|---|---|---|---|
| Nb_sessions_ouvertes | Entier | 0 | Nombre de sessions ouvertes par l'équipement à qui l'adresse correspondante a été allouée. |
| TTL | Entier | 60 heures | Durée d'allocation de l'adresse. |

Fig 3.4- Structure du pool d'adresses IPv4 géré par NAPT-PT

L'attribut *Nb_sessions_ouvertes* permet la désallocation de l'adresse lorsque le nombre de sessions ouverte passe à zéro. Ce nombre est incrémenté à chaque ouverture de session et décrémenté à chaque clôture de session.

L'adresse est allouée pour une durée limitée. Nous allons choisir ce temps suffisant pour éviter que les équipements se fassent éjecter alors qu'ils ont encore des paquets à transmettre. La clôture d'une session se faisant uniquement qu'à la suite d'une période d'inactivité.

Le statut alloué de l'adresse est scindé en deux : le statut préféré et le statut déprécié. Une adresse lors de l'allocation passe du statut disponible au statut préféré. Après le temps TTL, l'adresse passe du statut préféré au statut déprécié pour permettre à la machine de faire les rediffusions et de recevoir les réponses ou les acquittements.

Ensembe des sessions ouvertes

L'ensemble des sessions ouvertes *Session* est une liste dont la clé est composée des attributs Adr_v6_source et Adr_v4_destination.

| **Session** | | |
|---|---|---|
| **Attribut** | **Format** | **Description** |
| Adr v6 | x:x:x:x:x:x:x:x | Adresse IPv6 source |
| Adr v4 allouée | x.x.x.x | Adresse IPv4 attribués |
| | | |
| Adr v4 | x.x.x.x | Adresse IPv4 destination |
| TTL | Entier | Valeur courante du temporisateur représentant la durée de vie de l'adresse IPv4 attribuée |
| TTI | **Entier** | Valeur courante du temporisateur représentant le temps d'inactivité tolérable de la session |

Fig 3.4- Structure de la table des sessions ouvertes par NAT-PT

## Schéma états-transitions

Le schéma de la machine à état a été fait par un outil freeware [Libero]disponible sur Internet. Cet outil permet à partir d'un diagramme d'analyse générer le squelette d'un programme. Un schéma plus explicite équivalant se trouve en annexe 8.

```
After-Init:
    (--) Ok                                  -> Have-Get-Pckt-Feedback
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) Error                               ->
          + Set-Error-At-Program-Init
```

```
          + Terminate-The-Program

Have-Get-Pckt-Feedback:
    (--) Ok                                -> Have-Pckt-Session-Status
          + Check-And-Update-Sessions
    (--) Not-Found                         -> Have-Get-Pckt-Feedback
          + Pause
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) Error                             ->
          + Set-Get-Packet-Error
          + Terminate-The-Program

Have-Pckt-Session-Status:
    (--) Open-Session                      -> Have-Flag-Tcp
          + Check-Tcp-Flag
    (--) Closed-Session                    -> Have-Source-Address
          + Check-Source-Address

Have-Source-Address:
    (--) Ipv4-Source-Adr                   -> Have-Get-Pckt-Feedback
          + Discard-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) Ipv6-Source-Adr                   -> Have-V4-Layer-Status
          + Check-Ipv4-Equipment-Layer

Have-Flag-Tcp:
    (--) Flag-Fin                          -> Have-Get-Pckt-Feedback
          + Translate-The-Packet
          + Forward-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) Flag-Rst                          -> Have-Get-Pckt-Feedback
          + Discard-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet

Have-V4-Layer-Status:
    (--) V4-Address-Allocated              -> Have-Get-Pckt-Feedback
          + Open-New-Session
          + Translate-The-Packet
          + Forward-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) V4-Address-Not-Allocated          -> Have-Pool-Status
          + Check-Ipv4-Pool-Status

Have-Pool-Status:
    (--) V4-Address-Available              -> Have-Get-Pckt-Feedback
          + Allocate-Ipv4-Address
          + Open-New-Session
          + Translate-The-Packet
          + Forward-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) V4-Address-Not-Available          -> Have-Get-Pckt-Feedback
          + Discard-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
```

## Description des actions

Dans ce paragraphe, nous spécifions les différentes actions mises en évudence dans le schéma états-transitions de la figure 3.5.

### *Allocate-Ipv4-Address*

Le but de cette action est de faire basculer le statut de l'adresse IPv4 du pool du statut libre au statut alloué.

```
Pool[Adr_disponible].Adr_status ← 1
```

### *Check-And-Update-Sessions*

Le but de cette action est de tester si le paquet lu appartient à une session ouverte et de vérifier que les temporisateurs n'ont pas expirés.

```
Pour chaque S de la liste Session, faire
   Si {Paquet.Adr_source, Paquet.Adr_destination}⊂
      {S.Adr_v6, S.Adr_v4, S.Adr_v4_alloué} Alors
      w_adr_allouée ← oui
      Placer l'événement open_session
      Si temporisateur d'allocation expiré Alors
            Pool[S.Adr_v4_allouée].Adr_statut←DEPRECIE
      Si temporisateur d'inactivité expiré Alors
            Session ← Session-S
            Decrémenter de 1
            Pool[S.adr_v4_allouée].nb_sessions_ouvertes
            Si Pool[S.adr_v4_allouée].nb_sessions_ouvertes = 0 Alors
            Pool[S.adr_v4_allouée].Adr_statut ← DISPONIBLE
   Sinon
      Palcer l'événement closed_session
```

### *Check-Ipv4-Equipment-Layer*

Le but de cette action est de verifier si une adresse IPv4 est alloué à l'équipement IPv6 qui a émis le paquet.

```
Si w_adr_allouée Alors
      Placer l'événement v4_Address_Allocated
Sinon
      Placer l'événement v4_address_Not_allocated
```

### *Check-Ipv4-Pool-Status*

Le but de cette action est tester s'il existe une adresse disponible dans le pool d'adresses IPv4.

```
Tant qu'on n'est pas à la fin de la liste Et tant qu'on a pas trouvé
      Si P.Adr_status = 0 Alors
            Adr_disponible = P.Adr
            Placer l'événement v4_Address_Available
      Sinon
            Placer l'événement v4_Address_Not_Available
      Lire P
```

### *Check-Return-From-Get-Packet*

Le but de cette action est des tester le code retour lecture du paquet en entrée. Trois cas peuvent se présenter: on a erreur de lecture, il n'y a aucun paquet en entrée, on obtient en sortie le paquet lu.

```
Si code_retour = ok Alors
      placer l'événement ok
Si code_retour = End of file Alors
      placer l'événement Not_Found
Si code_retour = Erreur Alors
```

```
placer l'événement Error
```

*Check-Source-Address*

L'adresse source du paquet est soit une adresse IPv4 soit une adresse IPv6. Cette action permet permet de savoir de quelle nature de cette adresse source.

```
Si paquet.adr_source est du format x:x:x:x:x:x:x:x Alors
      Placer l'événement Ipv6_Source_Adr
Si paquet.adr_source est du format x.x.x.x Alors
                  Placer l'événement Ipv6_Source_Adr
```

***Check-Tcp-Flag***

Le but de cette action est de tester la valeur des bits FIN et RST du drapeau TCP.

```
Si paquet.Drap.RST = 1 Alors
                       Placer l'événement Flag_Rst
Sinon Si paquet.Drap.FIN = 1 Alors
      Placer l'événement Flag_Fin
```

***Discard-The-Packet***

Le but de cette action est d'arrêter le traitement du paquet courant. On lit directement le paquet suivant et le paquet courant n'est pas relayé par NAT-PT.

***Forward-The-Packet***

Le but de cette action est de relayer le paquet vers le destinataire.

***Get-Packet***

Le but de cette action est d'aller chercher le paquet à traiter sur la file d'entrée du routeur NAT-PT. On obtient soit un paquet IPv6 soit un paquet IPv4 au format présenté à la figure 3.3.

```
Code_retour ← Lire paquet
```

***Open-New-Session***

Le but de cette action est d'insérer une entrée dans la lliste des session ouvertes.

```
S.adr_v6 ← Paquet.adr_source
S.adr_v4_allouée ← Adr_disponible
S.adr_v4 ← Paquet.adr_destination
S.TTL = TTL
Armer les temporisateurs d'allocation et d'inactivité
Session = Session + S
Incrémenter de 1 Pool[adr_disponible].Nb_sessions_ouvertes
```

***Pause***

Etant donné que le processus NAT-PT fonctionne en continu, le but de cette action est de marquer une pause lorsque la file de paquets en entrée est vide. Le processus attend pendant un certain temps avant de refaire une lecture.

```
Pause(TEMPS_DE_PAUSE)
```

***Set-Error-At-Program-Init***

Le but de cette action est d'affecter une valeur adéquate à une variable de feedback.

```
feedback ← -1 (erreur dans l'initialisation du programme)
```

***Set-Get-Packet-Error***

***Terminate-The-Program***
Le but de cette action est d'arrêter le programme. Avant l'arrêt du programme on peut afficher des messages ou fermer des fichiers ouverts. On peut consulter le contenu de la variable feedback.

***Translate-The-Packet***
Le but de cette action est de traduire les paquets IPv4 en IPv6 et vice versa. Les détails de la traduction des paquets sont repris en annexe

### 3.3.2 NAPT-PT

## Ensemble des adresses et des numéros de port

L'ensemble des numéros de port et des adresses IPv4 correspondantes est une liste dont la clé d'accès aux différentes valeurs est la combinaison de l'adresse et du numéro de port.

| **Pool** | | | |
|---|---|---|---|
| **Attribut** | **Type/Format** | **Valeur par défaut** | **Description** |
| Adr | Format : x.x.x.x | | Adresse IPv4 |
| Num_port | Entier | | Numéro de port |
| Port_statut | Entier<br>0 :disponible<br>1 :préférée<br>2 : dépréciée | 0 | Statut de la paire adresses-numéro de port |

Figure 3.6 – Structure du pool d'adresses IPv4 géré par NAPT-PT

## Ensembe des sessions ouvertes

L'ensemble des sessions ouvertes *Session* est une liste dont la clé est composée des attributs Adr_v6_source et Adr_v4_destination.

| **Session** | | |
|---|---|---|
| **Attribut** | **Type/Format** | **Description** |
| Adr_v6 | x:x:x:x:x:x:x:x | Adresse IPv6 source |
| Port_v6 | Entier | Numéro de port de la source |
| Adr_v4_allouée | x.x.x.x | Adresse IPv4 attribués |
| Port_alloué | Entier | Numéro de port alloué |
| Adr_v4 | x.x.x.x | Adresse IPv4 destination |
| Port_v4 | Entier | Numéro de port destination |
| TTL | Entier | Valeur courante du temporisateur représentant la durée de vie de l'adresse IPv4 attribuée |
| TTI | ***Entier*** | Valeur courante du temporisateur représentant le temps d'inactivité tolérable de la session |

Figure 3.7 – Structure de la table des sessions ouvertes par NAPT-PT

## Correspondance IPv6 et IPv4 allouée

Le tableau de correspondance entre les adresses v6 et v4 permet de tenir à la disposition de la fonction de désallocation des adresses IPv4 le nombre de sessions ouvertes. Lorsque ce nombre atteint la valeur zéro, l'adresse peut être désallouée.

| V6 V4 | | |
|---|---|---|
| Attribut | Type/Format | Description |
| V6 adr | x:x:x:x:x:x:x | Adresse IPv6 de l'équipement |
| V4 adr | x.x.x.x | Adresse IPv4 allouée temporairement à l'équipement V6 |
| Nb session | Entier | Nombre de sessions ouvertes à un instant donné |

Figure 3.8 – Structure de la table du nombre de sessions ouvertes par un équipement IPv6

## Schéma états-transitions

```
After-Init:
    (--) Ok                                  -> Have-Get-Pckt-Feedback
         + Get-Packet
         + Check-Return-From-Get-Packet
    (--) Error                               ->
         + Set-Error-At-Program-Init
         + Teminate-The-Program

Have-Get-Pckt-Feedback:
    (--) Ok                                  -> Have-Pckt-Session-Status
         + Check-And-Update-Sessions
    (--) Not-Found                           -> Have-Get-Pckt-Feedback
         + Pause
         + Get-Packet
         + Terminate-The-Program
    (--) Error                               ->
         + Set-Get-Packet-Error
         + Terminate-The-Program

Have-Pckt-Session-Status:
    (--) Open-Session                        -> Have-Flag-Tcp
         + Check-Tcp-Flag
    (--) Closed-Session                      -> Have-Source-Address
         + Check-Source-Address

Have-Source-Address:
    (--) Ipv4-Source-Adr                     -> Have-Get-Pckt-Feedback
         + Discard-The-Packet
         + Get-Packet
         + Check-Return-From-Get-Packet
    (--) Ipv6-Source-Adr                     -> Have-V4-Layer-Status
         + Check-Ipv4-Equipment-Layer

Have-Flag-Tcp:
    (--) Flag-Fin                            -> Have-Get-Pckt-Feedback
         + Translate-The-Packet
         + Forward-The-Packet
         + Get-Packet
         + Check-Return-From-Get-Packet
    (--) Flag-Rst                            -> Have-Get-Pckt-Feedback
         + Discard-The-Packet
         + Get-Packet
         + Check-Return-From-Get-Packet
```

```
Have-V4-Layer-Status:
    (--) V4-Address-Allocated              -> Have-Port-In-Adr-Status
          + Check-Port-Availability-In-Address
    (--) V4-Address-Not-Allocated          -> Have-Pool-Status
          + Check-Ipv4-Pool-Status

Have-Pool-Status:
    (--) V4-Address-Available              -> Have-Get-Pckt-Feedback
          + Allocate-Ipv4-Address-And-Port
          + Open-New-Session
          + Translate-The-Packet
          + Forward-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) V4-Address-Not-Available          -> Have-Get-Pckt-Feedback
          + Discard-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet

Have-Port-In-Adr-Status:
    (--) Port-Available                    -> Have-Get-Pckt-Feedback
          + Allocate-Port-Number
          + Open-New-Session
          + Translate-The-Packet
          + Forward-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) Port-Not-Available                -> Have-Get-Pckt-Feedback
          + Discard-The-Packet
          + Get-Packet
          + Check-Return-From-Get-Packet
```

## Description des actions

### *Allocate-Ipv4-Address-And-Port*

Le but de cette action est d'allouer une adresse IPv4 et un numéro de port à un équipement IPv6 suite à l'initialisation d'une session.

```
Pool[adr_disponible, port_disponible].port_status ← 1
```

### *Allocate-Port-Number*

Le but de cette action est d'allouer un numéro de port à un équipement IPv6 à qui une adresse IPv4 est déjà allouée.

```
Pool[w_v4_adr, port_disponible].port_status ← 1
```

### *Check-And-Update-Sessions*

Comme dans le cas de NAT-PT, le but de cette action est de tester si le paquet lu appartient à une session ouverte et de vérifier que les temporisateurs n'ont pas expirés.

```
Pour chaque S de la liste Session, faire
   Si {Paquet.Adr_source, Paquet.Adr_destination,
      paquet.port_source, paquet.port_destination}⊂
      {S.Adr_v6, S.Adr_v4, S.Adr_v4_alloué
      S.port_v6, S.port_v4, S.port_alloué} Alors
      W_v4_adr_allouée←oui
      W_v4_adr←S.Adr_v4_allouée
      Placer l'événement open_session
      Si temporisateur d'allocation expiré Alors
```

```
            Pool[S.Adr_v4_alloué,S.port_alloué].port_statut←DEPRECIE
      Si temporisateur d'inactivité expiré Alors
            Session ← Session-S
            Decrémenter de 1
            V6_v4[S.Adr_v6,S.port_alloué].nb_session
            Si V6_v4[S.Adr_v6].nb_session = 0 Alors
            Pool[S.Adr_v4_alloué,S.port_alloué].Adr_statut ← DISPONIBLE
   Sinon
      Palcer l'événement closed_session
```

***Check-Ipv4-Equipment-Layer***

Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Check-Ipv4-Pool-Status***

Le but de cette action est de rechercher dans le pool un numéro de port libre

```
Tant qu'on n'est pas à la fin de la liste et tant qu'on a pas trouvé
      Si P.Port_status = 0 Alors
            Adr_disponible ← P.Adr
            Port_disponible ← P.Num_port
            Placer l'événement v4_Address_Available
      Sinon
            Placer l'événement v4_Address_Not_Available
      Lire P
```

***Check-Port-Availability-In-Address***

Etant donné une adresse IPv4 du pool, le but de cette action est de rechercher un numéro de port libre au sein de cette adresse.

```
Tant qu'on n'est pas à la fin de la liste et tant qu'on a pas trouvé
      Si P.Adr = w_v4_adr ET
            P.Adr_status = 0 Alors
            port_disponible ← P.num_port
            Adr_disponible ← P.Adr
            Placer l'événement Port_Available
      Sinon
            Placer l'événement Port_Not_Available
      Lire P
```

***Check-Return-From-Get-Packet***

Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Check-Source-Address***

Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Check-Tcp-Flag***

Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Get-Packet***

Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Open-New-Session***

```
S.adr_v6 ← Paquet.adr_source
S.port_v6 ← Paquet.port_source
S.adr_v4_allouée ← Adr_disponible
S.Port_alloué ← Port_disponible
```

```
S.adr_v4 ← Paquet.adr_destination
S.port_ destination ← Paquet.port_ destination
Armer le temporisateur d'allocation TTL
Armer le temporisateur d'inactivité TTI
Session ← Session + S
Incrémenter de 1 v6_v4[Paquet.adr_source].Nb_sessions
```

***Pause***
Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Set-Error-At-Program-Init***
Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Set-Get-Packet-Error***
Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Teminate-The-Program***
Cette action est identique à celle de même nom dans les spécifications de NAT-PT

***Translate-The-Packet***
Comme dans le cas de NAT-PT, la traduction des paquets se fait conformément au protocole SIIT en tenant compte de la traduction des numéros de port.

### 3.3.3 DSTM

Chaque client du domaine DSTM en communication garde en mémoire la valeur de l'adresse IPv4 qui lui a été allouée et la durée pendant laquelle cette adresse est valable. Il garde aussi l'adresse du TEP.

Le client DSTM maintient dans sa mémoire une correspondance entre l'adresse IPv6 du client et l'adresse IPv4 qui lui a été allouée. Il doit également savoir le temps d'allocation des adresses pour les désallouer en cas de dépassement du TTL.

Le TEP doit avoir une copie du tableau des adresses du serveur.

Pour la désallocation, nous avons besoin de deux champs techniques afin de stocker le contenu des deux temporisateurs.

| **Tablev6v4** | | |
|---|---|---|
| **Attribut** | **Type / Format** | **Description** |
| Adr ipv6 | x:x:x:x:x:x:x:x | Adresse IPv6 |
| Adr ipv4 | x.x.x.x | Adresse IPv4 |
| TTL | entier | Durée de vie de l'adresse |
| TTI | entier | Durée d'inactivité de la session |

Fig. –3.8 Format de la table des correspondances entre adresses IPv4 et adresses IPv6

| **Pool** | | |
|---|---|---|
| **Attribut** | **Type / Format** | **Description** |
| Adr v4 | x.x.x.x | Adresse IPv4 |
| Adr_status | Entier<br>0 :disponible | Statut de l'adresse IPv4 |

| | | |
|---|---|---|
| | 1 :préférée<br>2 : dépréciée | |

Fig. –3.9 Format du pool d'adresses IPv4 géré par DSTM

**Diagramme états-transitions**

```
After-Init:
    (--) Ok                                  -> Have-Get-Pckt-Feedback
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) Error                               ->
          + Set-Error-At-Program-Init
          + Terminate-The-Program

Have-Get-Pckt-Feedback:
    (--) Ok                                  -> Have-V4-Layer-Status
          + Check-Ipv4-Equipment-Layer
    (--) Not-Found                           ->
          + Set-Ended-Normally
          + Terminate-The-Program
    (--) Error                               ->
          + Set-Get-Packet-Error
          + Terminate-The-Program

Have-V4-Layer-Status:
    (--) V4-Address-Allocated                -> Have-Get-Pckt-Feedback
          + Encapsulate-The-Packet
          + Send-The-Packet-To-Tep
          + Decapsulate-The-Packet
          + Forward-The-Packet
          + Update-Table
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) V4-Address-Not-Allocated            -> Have-Server-Answer
          + Send-Request-To-Dstm-Server
          + Treat-The-Client-Request
          + Return-Sever-Answer
          + Check-Dstm-Server-Answer

Have-Server-Answer:
    (--) V4-Adr-Available                    -> Have-Get-Pckt-Feedback
          + Send-Table-Copy-To-Tep
          + Configure-Ipv4-Layer
          + Encapsulate-The-Packet
          + Send-The-Packet-To-Tep
          + Decapsulate-The-Packet
          + Forward-The-Packet
          + Update-Table
          + Get-Packet
          + Check-Return-From-Get-Packet
    (--) V4-Adr-Not-Available                -> Have-Get-Pckt-Feedback
          + Discard-The-Packet
          + Update-Table
          + Get-Packet
          + Check-Return-From-Get-Packet
```

**Description des actions**

***Check-Dstm-Server-Answer***

Le but de cette action est de vérifier si le serveur DSTM a alloué une adresse au client.

```
Si adr_libre <>"" Alors
      Placer l'événement v4_Adr_Available
Sinon
      Placer l'événement v4_Adr_Not_Available
```

***Check-Ipv4-Equipment-Layer***

Le but de cette action est de tester si l'équipement IPv6 possède une adresse IPv4 temporaire.

```
Si la clé paquet.adr_source existe dans tablev6v4 Alors
      Placer l'événement v4_Address_Allocated
Sinon
      Placer l'événement v4_Address_Not_Allocated
```

***Check-Return-From-Get-Packet***

Cette action est identique à celle de même spécifiée dans le cas de NAPT-PT.

***Treat-The-Client-Request***

Le but de cette action est de rechercher une adresse IPv4 disponible dans le pool

```
Tant qu'on a pas trouvé et tant qu'on est pas à la fin du Pool faire
      Si p.Adr_status = 0 Alors
            p.Adr_status ← 1
      adr_libre = p.adr_v4
      tablev6v4 ← tablev6v4 + t, avec
            t.adr_ipv6 ← paquet.adr_source
            t.adr_ipv4 ← adr_libre
            armer le temporisateur d'allocation TTL
            armer le temporisateur d'inactivité TTI
      lire p
```

***Update-Table***

```
Pour chaque élément t de la tablev6v4 faire
      Si Paquet.Adr_source = t.Adr_ipv6 Alors
            Armer le temporisateur d'inactivité
      Sinon
            Si temporisateur d'inactivité expiré Alors
                  Pool[t.Adr_ipv4] ← 2
            Si temporisateur d'allocation expiré Alors
                  Pool[t.Adr_ipv4].Adr_stattus ← 0
                  Supprimer la ligne courante de t
```

Les action non spécifiée sont soit équivalentes que celles du même nom dans le cas de NAT-PT et de NAPT-PT soit leur nom permet traduit ce qu'elles font.

## 3.6 Conclusions

Le passage de l'ancien protocole IPv4 vers le nouveau IPv6 doit se faire de manière progressive et c'est un réel défi de fournir un protocole IPv6 acceptable par les utilisateurs de l'Internet. Pour certains utilisateurs comme les ménages, les risques et les difficultés liés à la transition sont mineurs alors que les gestionnaires de grands réseaux doivent adapter plusieurs de leurs équipements afin qu'ils puissent s'intégrer dans le nouvel environnement.

Le problème que doit résoudre les équipements dans un environnement Internet de transition est de faire communiquer les ordinateurs implémentant le nouveau protocole IPv6 aux anciens ordinateur implémentant IPv4. Deux cas de figure se présentent, à savoir faire communiquer directement les équipement du nouveaux réseaux avec ceux de l'ancien d'une part, faire communiquer deux équipements de deux réseaux IPv6 éloignés et reliés par une infrastructure IPv4.

Les techniques de transitions spécifiées par l'IETF présentés au deuxième chapitre de ce travail permettent de résoudre ces problèmes. Parmi ces techniques, nous avons celles qui utilisent l'encapsulation des paquets IP et celles qui utilisent la traduction des adresses. L'encapsulation est utilisée dans la communication par tunnel. Celle-ci nécessitent une configuration manuelle des équipements situés de part et d'autre du tunnel.

Parmi les techniques de transition proposées, nous avons choisi d'analyser dans les détails les protocoles NAT-PT, NAPT et DSTM afin de déceler les difficultés qu'on peut rencontrer dans leur implémentation.

NAT-PT dans sa conception est assez simple et pourrait facilement être mis en œuvre. La marche à suivre utilisée pour le traitement des paquets serait la plus facile à implémenter. Les différentes actions à effectuer étant limitées. L'équipement IPv6 ne demande pas une adresse IPv4 à un serveur comme dans le cas de DSTM.

Le protocole NAPT-PT apporte une amélioration par rapport à NAT-PT dans la mesure où plusieurs équipements peuvent utiliser une même adresse IPv4. Cette amélioration est possible avec une gestion plus intelligente du pool d'adresses IPv4. En effet une mauvaise allocation des numéros de port peut entraîner qu'un ordinateur ne puisse pas ouvrir de session malgré le fait qu'il possède une adresse parce que tous les numéros de port au sein de l'adresse sont alloué. La seule façon dans ce cas de satisfaire une telle demande est d'allouer une nouvelle adresse IPv4 à l'équipement.

La technique DSTM semble la plus lourde à mettre en œuvre dans la mesure où elle fait intervenir plusieurs entités physiques séparées. On peut imaginer un regroupement du TEP et du serveur DSTM au sein d'un même équipement. Mais cette solution ne résout pas le problème de la surcharge du réseau par l'envoie des requêtes et le renvoie des réponses de ces requêtes.

DSTM demande également une gestion des communications entre processus s'exécutant sur des ordinateurs différents. Donc une mise à jour de tous les équipements appartenant au domaine DSTM est nécessaire. Ce qui n'est pas facile pour des réseaux de grande importance.

Au niveau des implémentations des trois techniques, la gestion des temporisateur nécessite une attention particulière dans la mesure où il faut gérer deux temporisateur au niveau de chaque session.

Pour confronter ces remarques sus-présentées des différentes techniques de transition, une évaluation quantitative des performances peut être faite à partir d'une trace réelle de trafic. Les paramètres intéressants à mesurer sont: le nombre d'adresses alloué, le nombre de paquets transférés vers les destinataires, le nombre de paquets détruits parce que le pool d'adresses et de numéros de port est vide. Ce paramètres doivent être comparés au nombre de paquets reçus en entrée. Le protocole qui peut transférer un nombre supérieur de paquet par rapport aux autres en allouant un nombre limité d'adresses IPv4 aux équipements IPv6 est le plus performant.

L'utilisation du timestamp de la trame peut permettre de dimensionner le temporisateur de durée de vie et le temporisateur d'inactivité. En effet une session est inactive lorsque la différence entre le timestamp du paquet courant et celui du dernier paquet traiter dans la session est supérieur à une certaine valeur timeout. La temps relatif à la durée de vie d'une adresse est écoulé lorsque la différence entre le timestamp du paquet courant et le timestamp du paquet qui a ouvert la session est nulle.

En attendant, nous pouvons dire que la transition de IPv4 vers IPv6 va coûter de l'argent, elle va prendre une décennie, elle va demander un remplacement des logiciel et une formation du personnel de gestion des réseau.

# Bibliograohie

*Livres*

[Ciz99] Cizault, G., IPv6 Théorie et pratique, O'Reilly, 1999.

[Hui96] Huitema, C., IPv6: The New Internet Protocol. Prentice-Hall, Upper Saddle River, NJ: 1996.

[Rif98] Rifflet, J.M., La programmation sous Unix,3e édition, Edicscience, 1998.

[Tan97] Tanenbaum A., Réseaux, 3e édition, Prentice Hall, 1997.

*RFCs*

[RFC 826] Plummer D.,C., An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware, November 1982.

[RFC 1191] Mogul J., Deering S., Path MTU Discovery, November 1990.

[RFC 1519] Fuller V., Li T., Yu J., Varadhan K., Classless Inter-Domain Routing (CIDR):an Address Assignment and Aggregation Strategy, September 1993.

[RFC 1886] Thomson S., Huitena C., DNS Extensions to support IP version 6, December 1995.

[RFC 2026] Bradner S., The Internet Standards Process -- Revision 3, October 1996.

[RFC 2113] D. Katz D., IP Router Alert Option, February 1997.

[RFC 2373] Hinden R., Deering S., IP Version 6 Addressing Architecture, July 1998.

[RFC 2374] Hinden R., O'Dell M.,Deering S., An IPv6 Aggregatable Global Unicast Address Format, July 1998.

[RFC2460] Deering S., Hinden R., Internet Protocol, Version 6 (IPv6) Specification, December 1998.

RFC 2474] Nichols K., Blake F., Black D., Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, December 1998.

[RFC 2529] Carpenter B., Jung C., Transmission of IPv6 over IPv4 Domains without Explicit Tunnels, March 1999.

[RFC 2765] Nordmark, E., Stateless IP/ICMP Translation Algorithm (SIIT), February 2000.

[RFC 2766] G. Tsirtsis G., Srisuresh P., Network Address Translation - Protocol Translation (NAT-PT), February 2000.

[RFC 3053] Durand A., Fasano, P., Guardini I., Lento D., IPv6 Tunnel Broker, January 2001.

*Internet Drafts*

Bound J., Toutain L., Dupont F., Afifi O., Durand A., Dual Stack Transition Mechanism (DSTM)

**Annexe 1 – Répartition des adresses IPv6**

| Préfixe | Attribution | Fraction de l'espace d'adressage |
|---|---|---|
| 0::/8 | Réservé (compatible IPv4) | 1/256 |
| 100::/8 | non attribué | 1/256 |
| 200::/7 | Adresse NSAP OSI | 1/128 |
| 400::/7 | Adresse Netware IPX Novell | 1/128 |
| 600::/7 | Non attribué | 1/128 |
| 800::/5 | Non attribué | 1/32 |
| 1000::/4 | Non attribué | 1/16 |
| 2000::/3 | Adresse affecté à un fournisseur de service (Aggregation-based unicast address) | 1/8 |
| 4000::/3 | Adresse affectée à s (Geographic-based unicast address) | 1/8 |
| 6000::/3 | Non attribué | 1/8 |
| 8000::/3 | Adresse affectée à caractère géographique (Geographic-based unicast address | 1/8 |
| A000::/3 | Non attribué | 1/8 |
| C000::/3 | Non attribué | 1/8 |
| E000::/4 | Non attribué | 1/16 |
| F000::/5 | Non attribué | 1/32 |
| F800::/6 | Non attribué | 1/64 |
| FC00::/7 | Non attribué | 1/128 |
| FE00::/9 | Non attribué | 1/512 |
| FE80::/10 | Adresse de lien local | 1/1024 |
| FEC0::/10 | Adresse de site local | 1/1024 |
| FF00::/8 | Adresse de diffusion multidestinataire | 1/256 |

**Annexe 2 – Répartition des adresses IPv6**

| En-tête d'extension | Valeur du champ en-tête suivant de l'en-tête précédent | Description |
|---|---|---|
| Proche-en-proche | 0 | Informations destinées aux routeurs |
| Routage | 43 | Route à suivre complètement ou partiellement |
| Fragmentation | 44 | Gestion des fragments de datagramme |
| Authentification | 51 | Vérification de l'identité de l'émetteur |
| Confidentialité | 50 | Information sur le chiffrement des données |
| Fin des en-têtes | 59 | |
| Destination | 60 | Information additionnelle pour la destination |

**Annexe 3 –Les options de l'en-tête d'extension proche-rn-proche**

00: le routeur ignore l'option.

01: le routeur rejette le paquet.

10: le routeur rejette le paquet et retourne un message ICMPv6 d'inaccessibilité.

11: le routeur rejette le paquet et retourne un message ICMPv6 d'inaccessibilité si l'adresse de destination n'est pas multicast.

Le troisième bit du type indique que le routeur peut modifier le contenu des options -valeur à 1- ou non -valeur à 0-.

Les quatre options de l'en-tête proche-en-proche sont:

- Pad 1 –type 0–. Cette option est utilisée pour introduire un octet de bourrage.
- Pad n –type 1–. Cette option est utilisée pour introduire plus de deux octets de bourrage. Les options de bourrage permettent d'optimiser le traitement des paquets.
- Jumbogramme –type 194 ou 0xC2–. Cette option est utilisée quand le champ longueur de données du paquet IPv6 n'est pas suffisant pour coder la taille du paquet. Cette option est essentiellement prévue pour la transmission à grand débit entre deux équipements. Si l'option jumbogramme est utilisée, le champ longueur des données utiles dans l'en-tête IPv6 vaut 0. Le type correspondant à cette option commence par la séquence binaire 11. Ce qui permet aux routeurs ne traitant pas les jumbogrammes d'en informer la source. Celle-ci pourra réémettre l'information sans utiliser cette option.
- L'option Router Alert a aussi été définie dans IPv4 –RFC 2113–. Elle demande à un routeur d'examiner le contenu des données qu'il relaie. En principe le processus de relais – recopier le paquet sur une interface de sortie en fonction de l'adresse du destinataire et des tables de routage– doit être le plus rapide possible. Mais pour des protocoles comme la gestion des groupes multicast avec ICMPv6, les routeurs intermédiaires doivent prendre en compte les données. L'émetteur envoie les données à la destination, mais s'il précise

l'option Router Alert alors les routeurs intermédiaires vont analyser les donnée, voire modifier leur contenu avant de relayer le paquet. Ce mécanisme est efficace puisque les routeurs n'ont pas à analyser le contenu de tous les paquets d'un flux. Le type de l'option vaut 5. Il commence par la séquence binaire 00 puisqu'un routeur qui ne connaît pas cette option doit relayer le paquet sans le modifier.

Le champ valeur de l'option contient:

0 pour les messages ICMPv6 de gestion des groupes multicast;

1 pour les message RSVP;

2 pour les réseaux actifs.

Les autres valeurs sont réservées.

**Annexe 4 - Valeurs des champs type et code d'ICMPv6**

| Type | Code | Nature |
|---|---|---|
| | | *Gestion des erreurs* |
| 1 | | Destination inaccessible |
| | 0 | Aucune route vers la destination |
| | 1 | Communication avec le destination administrativement interdite |
| | 2 | La destination n'est pas un voisin |
| | 3 | Adresse inaccessible |
| | 4 | Numéro de port inaccessible |
| 2 | | Paquet trop grand |
| 3 | | Temps dépassé |
| | 0 | Limite du nombre de sauts atteinte |
| | 1 | Temps de ré assemblage dépassé |
| 4 | | Erreur de paramètre |
| | 0 | Champ d'en-tête erroné |
| | 1 | Champ d'en-tête suivant non reconnu |
| | 2 | Option non reconnue |
| | | *information* |
| 128 | | Demande d'écho |
| 129 | | Réponse d'écho |
| 130 | | Demande de gestion de groupe multicast |
| 131 | | Rapport de gestion de groupe multicast |
| 132 | | Réduction d'un groupe multicast |

| | | |
|---|---|---|
| | | *Découverte des voisins* |
| 133 | | Sollicitation du routeur |
| 134 | | Annonce du routeur |
| 135 | | Sollicitation d'un voisin |
| 136 | | Annonce d'un voisin |
| 137 | | redirection |
| | | *Renumérotation des routeurs* |
| 138 | | Renumérotation des routeurs |
| | 0 | Commande |
| | 1 | Résultat |
| | 255 | Remise à zéro du numéro de séquence |

**Annexe 5 – Types de message DHCPv6**

- Sollicitation DHCP : message émis vers un serveur ou un relais DHCP. Un client émet un tel message pour obtenir l'adresse d'un serveur DHCP. La machine utilise comme adresse destinataire FF02::1:2 qui est l'adresse multicast des agents DHCP.
- Annonce DHCP : message émis en réponse à un message de sollicitation DHCP. Il donne l'adresse IPv6 d'un serveur DHCP.
- Requête DHCP : message émis par un client pour demander les paramètres de configuration au serveur.
- Réponse DHCP : réponse émise par le serveur suite à une demande du client.
- Libération DHCP : demande du client de libérer de libérer des ressources allouées par le serveur.
- Reconfiguration DHCP : message émis par le serveur pour informer le client qu'il a de nouvelles informations de configuration. Le client doit alors commencer une nouvelle transaction pour acquérir des informations.

**Annexe 6 –Encapsulation IPv6 en IPv4**

Version: 4

Lg_ent –en mots de 32 bits–: 5 car il n'y a pas d'options IPv4

Type de service: 0

Longueur totale: taille du paquet IPv6 plus taille de l'en-tête IPv4

Identification: généré comme dans tous les paquets IPv4 transmis dans le système.

Drap: pour le bit *DF*, voir le paragraphe *Tunnel MTU et fragmentation*; allumer le bit *MF* si nécessaire.

Dep_fragment: selon la fragmentation

TTL: selon l'implémentation

Protocole: 41

Checksum: somme de contrôle calculée du paquet IPv4

Adresse source: adresse IPv4 de l'interface de sortie du nœud d'entrée du tunnel

Adresse Destination: adresse IPv4 du nœud de sortie du tunnel.

**Annexe - 7**

Traduction en-tête IPv4 vers en-tête IPv6

Si le paquet IPv4 peut être fragmenté –bit *DF* non positionné– et si le paquet IPv6 obtenue a une taille de plus de 1280 octets, alors celui-ci doit être fragmenté avant toute traduction d'en-tête.

Si par contre le bit DF est positionné, et si le paquet n'est pas un fragment –bit *MF* non positionné et le champ *dep_fragment* étant à 0– alors l'en-tête de fragmentation ne doit pas être ajoutée.

La traduction de l'en-tête IPv4 vers l'en-tête IPv6 se fait de façon suivante:

*Version*: 6

*Classe de trafic*: copie du champ *Type de service* par défaut. Une implémentation du traducteur doit pouvoir ignorer le champ *ToS* et garnir le champ *Classe de trafic* de zéros.

*Identificateur de flux*: tous les bit à zéro.

*Longueur des données*: longueur totale du paquet IPv4 moins la taille de l'en-tête et des options si elles existent.

*En-tête suivant*: copie du champ protocole de l'en-tête du paquet IPv4.

*Nombre de saut*: copie du champ *TTL* de l'en-tête IPv4. Si le traducteur est un routeur, celui-ci décrémente soit la valeur de *TTL* de l'en-tête IPv4 –avant la traduction– ou la valeur du champ nombre de sauts de l'en-tête IPv6 –après traduction–. Le traducteur doit aussi vérifier si la valeur de ce champ est égale à zéro. Dans ce cas, un message ICMPv4 ou ICMPv6 est envoyé.

*Adresse source*: adresse IPv4-mappé.

*Adresse destination*: adresse IPv4-traduite.

Les options si elles sont présentes ne sont pas traduites.

Si le bit *DF* n'est pas positionné ou si le paquet est un fragment, la traduction se fait comme nous l'avons présenté ci-dessus avec les exception suivantes:

Pour les champs de l'en-tête IPv6 ,

-*Longueur des données* : taille totale du paquet IPv4 moins la taille des en-têtes –en-tête IPv4 et en-tête de fragmentation si elle est présente–.

-*En-tête suivant* : en-tête du fragment (44).

Pour les champs de l'en-tête du fragment :

-*En-tête suivant* : copie du champ protocole de l'en-tête IPv4.

-*Place du fragment*: copie du champ *dep_fragment* de l'en-tête IPv4.

-Bit *M* : copie de l'en-tête IPv4.

-*Identification* : les 16 bits de poids faible sont copiés à partir du champ *Identification* de l'en-tête IPv4 et les 16 bits de poids fort sont mis à 0.

Traduction des en-têtes ICMPv4 en en-têtes ICMPv6

Tous les message ICMP qui doivent être traduits nécessitent une mise à jour des champs *Checksum* et *Type* de l'en-tête. Pour les messages d'erreur, l'en-tête IP incluse doit aussi être traduit.

Messages d'information :

- Demande et réponse d'écho –*Type* 15 et *Type* 16– : ajuster le Type à 128 et 129 respectivement et ajuster le *Checksum* ICMP.

- Demande et réponse d'information –*Type* 8 et *Type* 0–: obsolètes dans ICMPv6 donc détruits par le traducteur.

- Timestamp et réponse de Timestamp –*Type* 13 et *Type* 14–: obsolètes dans ICMPv6 donc détruits par le traducteur.

- Demande et réponse de masque d'adresse –*Type* 17 et *Type* 18–: obsolètes dans ICMPv6 donc détruits par le traducteur.

- Annonce d'un routeur –*Type* 9–: message d'un saut donc détruits par le traducteur.

-Sollicitation d'un routeur –*Type* 10–: message d'un saut donc détruits par le traducteur.

- Tous les autres messages ICMPv4 de type inconnu sont détruits.

Messages IGMP:

Tous les messages IGMP sont détruits car se sont des messages d'un saut.

Messages d'erreurs ICMPv4:

-Destination inaccessible –*Type* 3–: pour tous les messages non cités ci-dessous, utiliser la valeur 1 pour le *Type*.

Code 0, 1 –réseau ou hôte inaccessible–: Code = 0 –aucune route vers la destination–

Code 2 –protocole inaccessible–: traduire en un message ICMPv6 erreur de paramètre de *Type* 4 et de *Code* 1. Faite pointer le champ *Pointeur* vers le champ *en-tête suivant*.

Code 3 –port inaccessible–: Code = 4 –port inaccessible–

Code 4 –bit *DF* allumé et nécessité de fragmenter–: traduire en un message ICMPv6 paquet trop grand *Type* 2 et *Code* 0. Le champs *MTU* doit être ajusté en tenant compte de la différence de taille entre les en-têtes IPv4 et les en-tête IPv6.

Code 5 –échec de routage par la source–: *Code* = 0 –aucune route vers la destination–

Code 6, 7: *Code* = 0 –aucune route vers la destination–

Code 8: *Code* = 0 –aucune route vers la destination–

Code 9, 10 –communication avec la destination administrativement interdite–: Code = 1

Code 11, 12: *Code* = 0 –aucune route vers la destination–

- Redirection –*Type* 5–: message d'un saut donc détruit par le traducteur.

- Source Quench –Type 4–: obsolète dans ICMPv6 dont détruit.

- Temps dépassé –Type 11–: *Type* = 3 et *Code* reste inchangé

Traduction des messages d'erreur ICMPv4 en ICMPv6

Il existe des différences entre le format des messages d'erreurs ICMPv4 et ceux des messages ICMPv6. En plus, les messages ICMP d'erreur contiennent l'en-tête IP du paquet en erreur. Cette en-tête IP doit être traduite comme une en-tête IP classique. La traduction de ce paquet en erreur entraîne un changement de la taille du datagramme ; de ce fait, la charge utile de l'en-tête IPv6 extérieure doit être mise à jour.

Quand un paquet IPv4 doit-il être traduit en IPv6?

Le traducteur est supposé savoir les adresses du pool IPv4 utilisées pour représenter les nœuds locaux IPv6-only. Si l'adresse IPv4 de destination appartient cet l'ensemble, alors le paquet doit être traduit en IPv6.

Traduction des en-têtes IPv6 en en-têtes IPv4

Si l'en-tête de fragmentation n'est pas présente, l'en-tête IPv4 est garnie de la manière suivante :
Version : 4
Longueur de l'en-tête : 5 –sans options–
Type de service : par défaut, une copie du champs classe de trafic dans IPv6. Dans certaine implémetations où les bit du sous champ DiffServ ont l'ancienne sémantique, le traducteur doit être à mesure d'ignorer le champ classe de trafic et mettre zéro dans le champ ToS.
Longueur totale : valeur du champ longueur des données de IPv6 plus taille de l'en-tête IPv4
Identification : tous des zéros
Drap : le bit MF est mis à zéro et le bit DF est mis à un
Dep_fragment : tous des zéros.

Durée de vie :copie du champ Nombre de saut de l'en-tête IPv6. Si le traducteur est un routeur, celui-ci décrémenter la valeur de ce champ avant traduction. Une valeur nulle de champ entraîne l'envoie d'un message ICMPv6.
Protocole : copie du champ en-tête suivant de l'en-tête IPv6.
Checksum : calculée une fois l'en-tête IPv4 créée.
Adresse source :Si l'adresse source est une adresse IPv4 traduite, alors les 32 bits de poids faible constituent l'adresse source IPv4. Sinon remplir l'adresse source par la valeur 0.0.0.0.
Adresse Destination : un paquet IPv6 traduit a pour adresse source une adresse IPv4-mappée. Dans ce cas ses 32 bits de poids faible sont copiés dans le champ adresse destination de l'en-tête IPv4.
Les en-tête des options suivantes sont ignorées par le traducteur :proche-en-proche, Destination, routage avec le champ segments restants à 0. Cependant les champs lg.extension et protocole doivent être mis à jour.
Si une en-tête d'extension de routage avec un champ segments restant égale à zéro, alors le paquet n'est pas traduit et un message ICMPv6 d'erreur de type 4, de code 0, avec le champ Pointeur indiquant le champ segments restants est envoyé à la source.
Si le paquet IPv6 contient une en-tête de fragmentation, alors les champs sont garnis de la façon suivante :
Longueur totale : longueur des données de l'en-tête IPv6, moins 8 pour l'en-tête du fragmentation, plus taille de l'en-tête IPv4.
Identification : copie des 16 bits du champ identification de l'en-tête de fragmentation.
Drap : le bit MF est copié à partir du bit M de l'en-tête d'extension IPv6. Le Bit DF est mis à zéro autorisant ainsi aux routeurs IPv4 de fragmenter le paquet.
Dep_fragment : copie du champ place du fragment de l'en-tête d'extension de fragmentation.
Protocole : copie du champ en-tête suivant de l'en-tête de fragmentation.

**Annexe 8 - Traits de comparaison de IPv6 et de IPv4**

| Elément de comparaison | IPv4 | IPv6 |
|---|---|---|
| Adresse | - 32 bits (4 octets) | - 128 bits (16 octets) |
| Espace d'adressage | - 10E9 adresses possibles | - 10E38 adresses possibles |
| En-tête du datagramme | - Taille variable, perte de temps dans la gestion | - Taille fixe (40 octets), gestion plus efficace |
| Champs spéciaux dans l'en-tête | - plusieurs, pas toujours supporté par certaines implémentations ce qui diminu la performance | - éliminés ou remplacés par d'autres pour augmenter la performance |
| Taille du paquet | - 65536 octets maximum | - paquet normal plus de 65536 octets<br>- utilisation des jumbogrammes |
| Allocation d'adresses | - par les classes de réseau A, B, C (réseaux larges, moyens, petits)<br>- pas de hiérarchie, augmentation des tables de routage<br>- utilisation locale limitée au lien<br>- expansion impossible | - compatibilité avec IPv4<br>- hiérachisé par registre, par provider et par sous-réseaux<br>- hiérarchisée par régions géographiques<br>- utilisation locale par le lien ou par le site<br>- plus de 70 % des adresses réservé pour l'expansion |
| Notation d'adresse | - décimale pointée | - hexadécimale avec deux points et avec abréviations; IPv4 est un cas particulier |
| Types d'adresses | - point à point<br>- broadcast local; mulicast limité<br>- anycast (expérimental) | - Multicast par lien, par site, par organisation, par groupe<br>- Anycast |
| Fragmentation | - Plusieurs phases de fragmentation, faite par les routeurs, impacte sur la performance | - faite une seule fois, par l'hote, après découverte du MTU, performance des routeurs améliorée |
| Qualité de service | - définie mais pas utilisée | - étiquete de flow<br>- priorité<br>- supporte le tranfert de données temps réel et multimédia |
| Sécurité | - limitée, pas d'authentification ou d'encryption au niveau IP (dépendance des protocoles de niveau supérieur donc vulnérable) | - authentification (validation de l'origine du paquet)<br>- encryption (privacy of content)<br>- demande une gestion centralisé de la distribution des clés |

| | | |
|---|---|---|
| Gestion de la Configuration | - manuelle, investissement en temps même pour un petit réseau<br>- résolution d'adresse des stations localement<br>- Grande dépendance des routes par défaut | - configuration automatique des adresses lien-local basée sur les adresses physiques<br>- configuration sans état des réseaux simples<br>- supporte des stations sans disque<br>- intervention humaine réservée au environnements complexes<br>- l'algorithme de découverte des voisins construit les routes |
| Gestion du routage | - BGP-4 entre sous-domaines<br>- Utilise TCP<br>- Conçu pour les adresses 32 bits<br>- Une seule famille d'adresse<br>- Utilisation de grandes tables<br>- OSPF, RIP à l'intérieur de sous-domaines | - IDRP entre sous-domaines<br>- Basé sur les datagrammes IP<br>- Adapté aux adresses 128 bits<br>- Plusieurs type d'adresses<br>- Tables plus agrégées<br>- Mise à jour de OSPF et de RIP mais similaire |

Annexe 9 – Diagramme états-transitons

-schema=lrschema.pl

## After-Init

| | | |
|---|---|---|
| Ok | Get-Packet<br>Check-Return-From-Get-Packet | Have-Get-Pckt-Feedback |
| Error | Set-Error-At-Program-Init<br>Terminate-The-Program | |

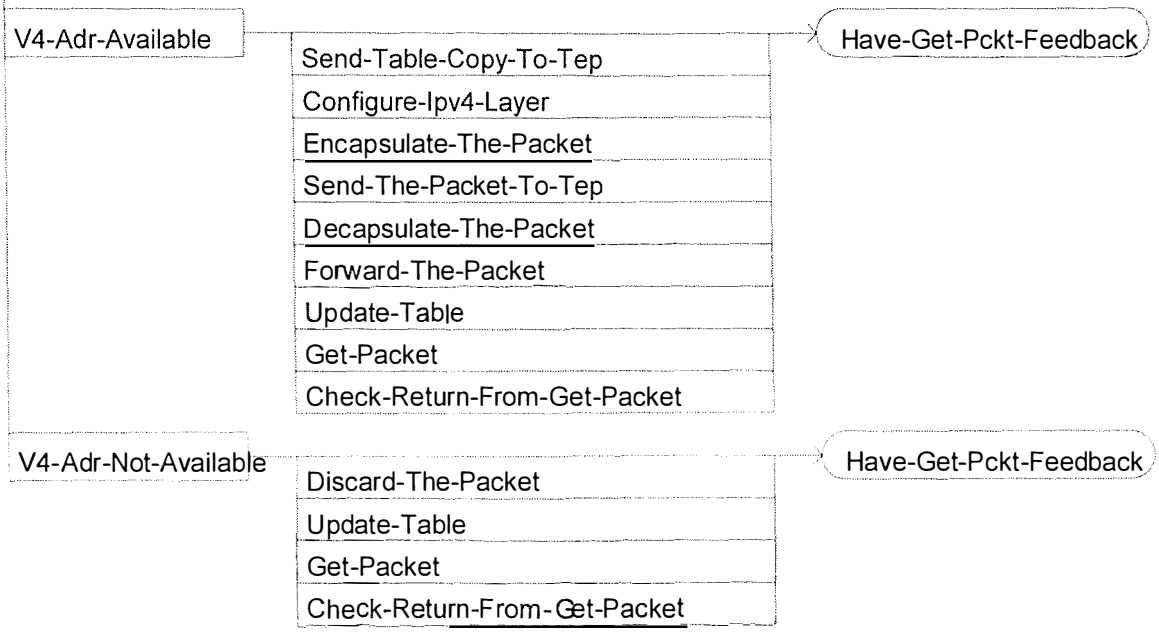
## Have-Get-Pckt-Feedback

| | | |
|---|---|---|
| Ok | Check-Ipv4-Equipment-Layer | Have-V4-Layer-Status |
| Not-Found | Set-Ended-Normally<br>Terminate-The-Program | |
| Error | Set-Get-Packet-Error<br>Terminate-The-Program | |

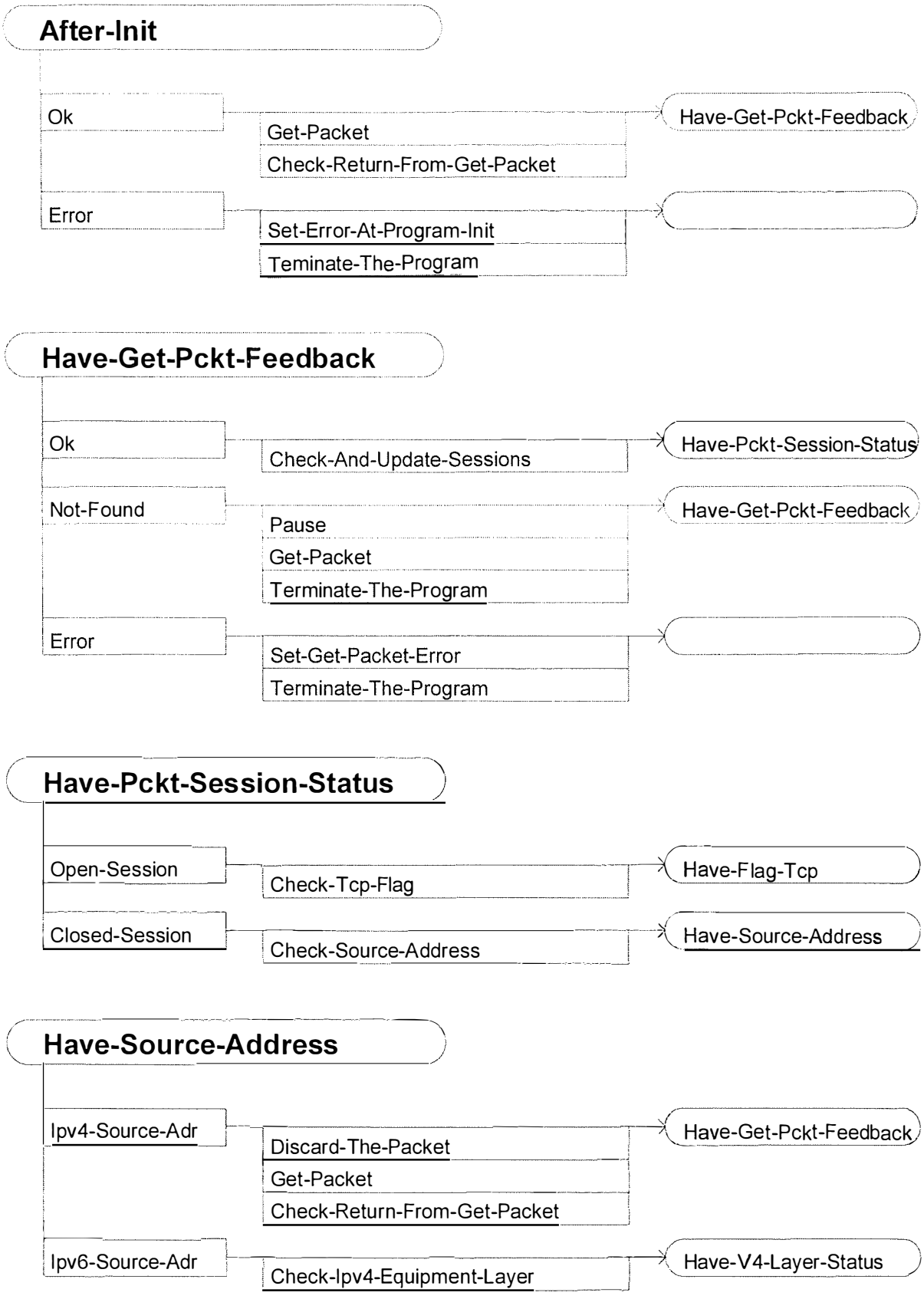
## Have-V4-Layer-Status

| | | |
|---|---|---|
| V4-Address-Allocated | Encapsulate-The-Packet<br>Send-The-Packet-To-Tep<br>Decapsulate-The-Packet<br>Forward-The-Packet<br>Update-Table<br>Get-Packet<br>Check-Return-From-Get-Packet | Have-Get-Pckt-Feedback |
| V4-Address-Not-Allocated | Send-Request-To-Dstm-Server<br>Treat-The-Client-Request<br>Return-Sever-Answer<br>Check-Dstm-Server-Answer | Have-Server-Answer |

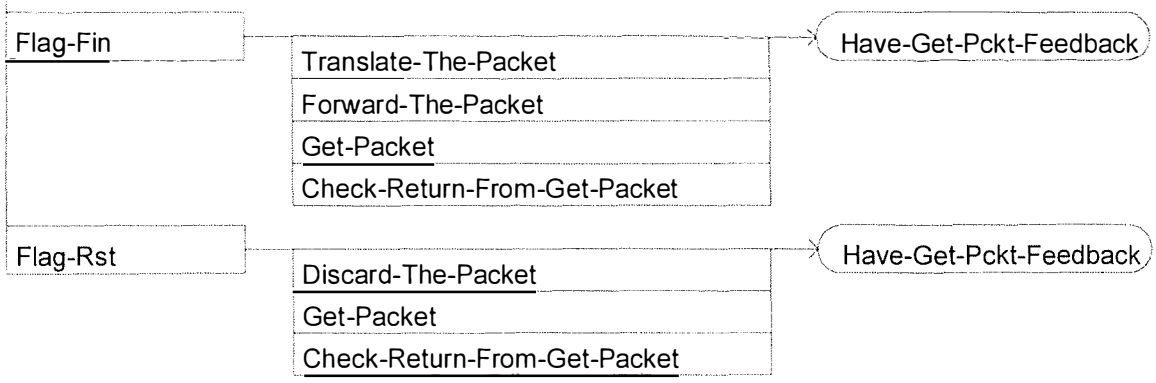
## Have-Server-Answer


V4-Adr-Available
Send-Table-Copy-To-Tep
Configure-Ipv4-Layer
Encapsulate-The-Packet
Send-The-Packet-To-Tep
Decapsulate-The-Packet
Forward-The-Packet
Update-Table
Get-Packet
Check-Return-From-Get-Packet
Have-Get-Pckt-Feedback
V4-Adr-Not-Available
Discard-The-Packet
Update-Table
Get-Packet
Check-Return-From-Get-Packet
Have-Get-Pckt-Feedback

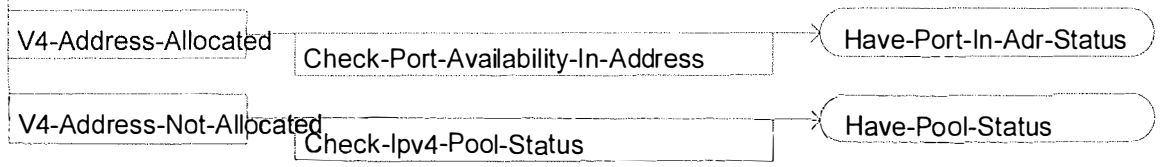
-schema=lrschema.pl



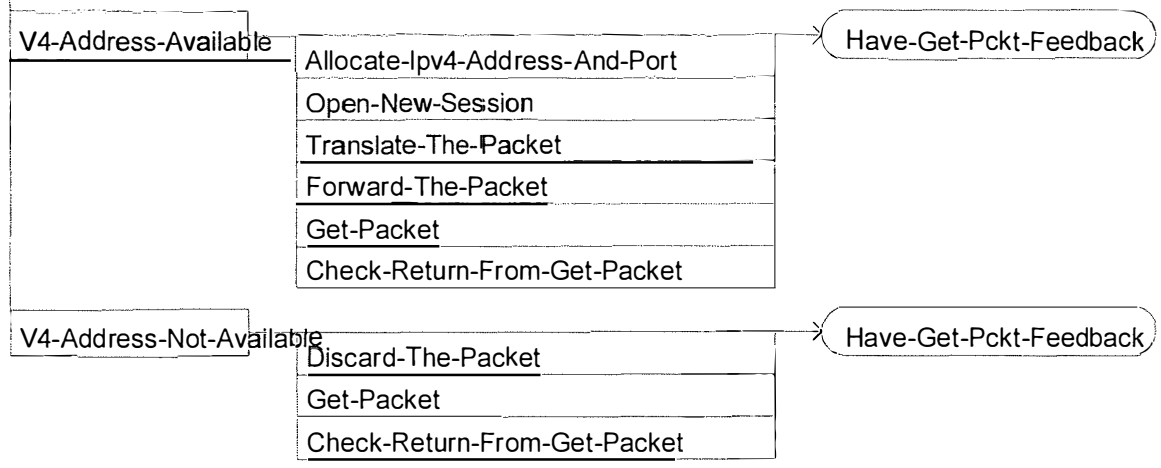
## Have-Flag-Tcp



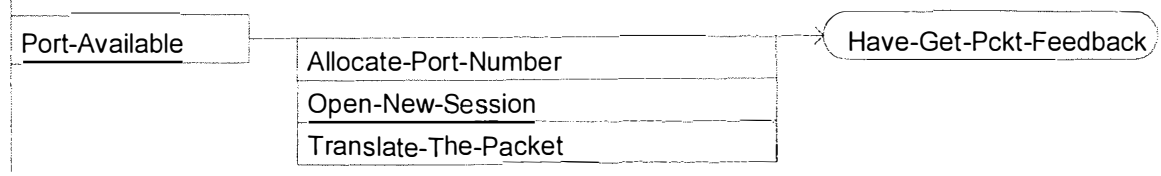

## Have-V4-Layer-Status

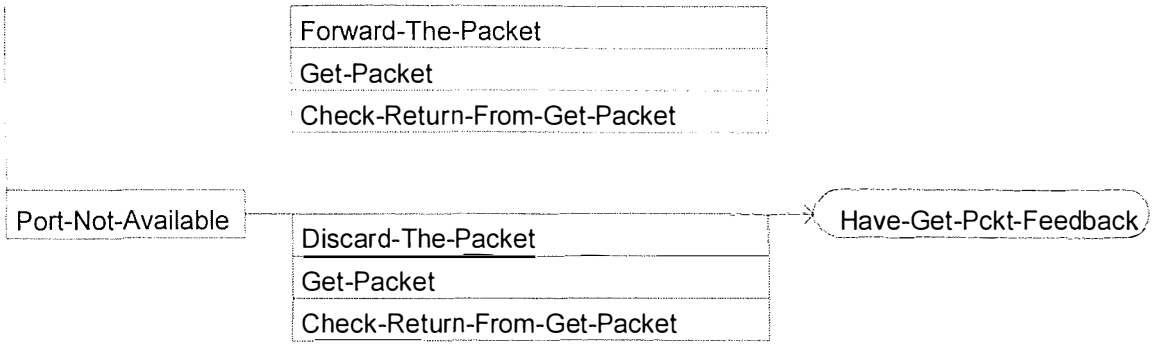



## Have-Pool-Status

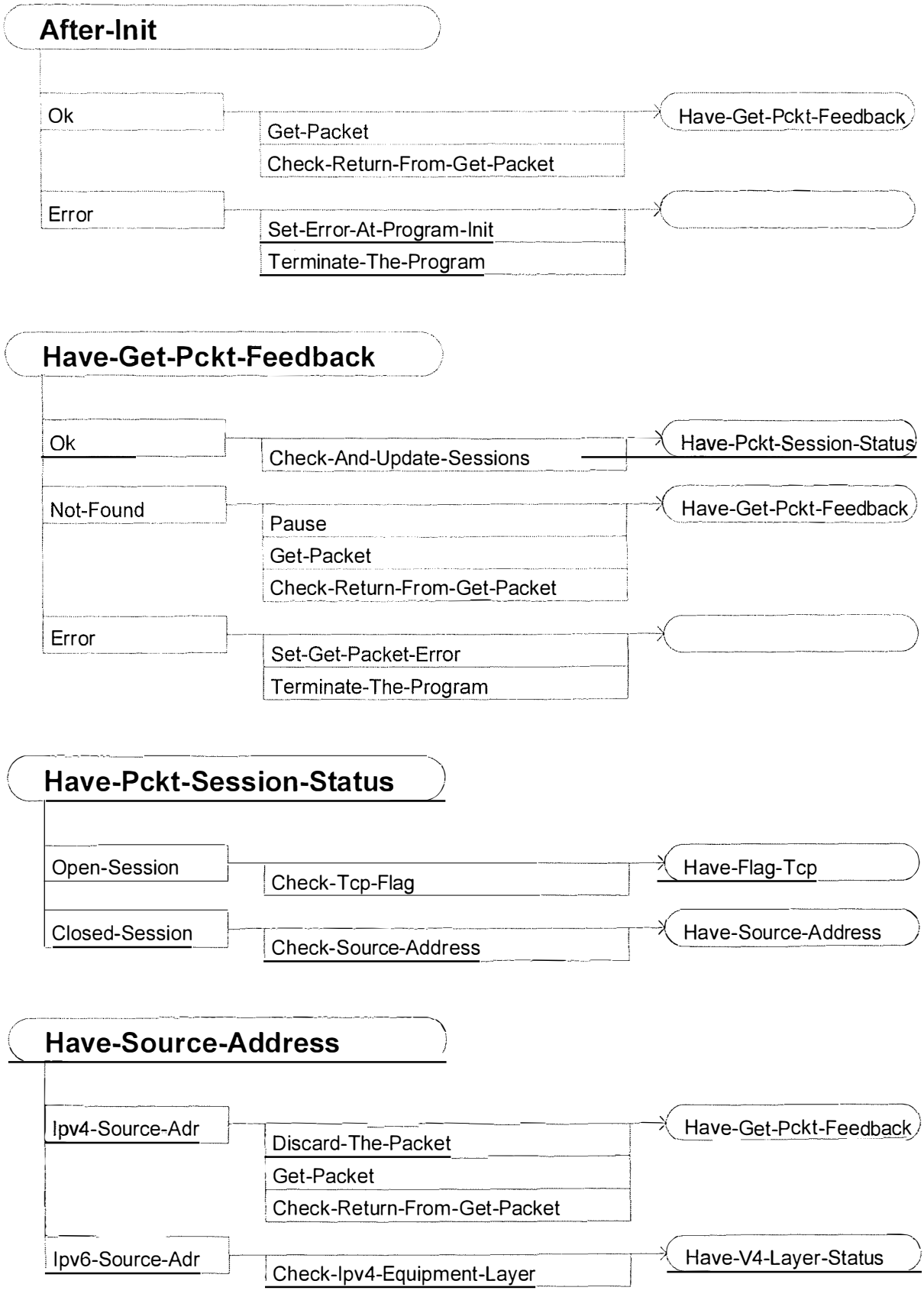



## Have-Port-In-Adr-Status

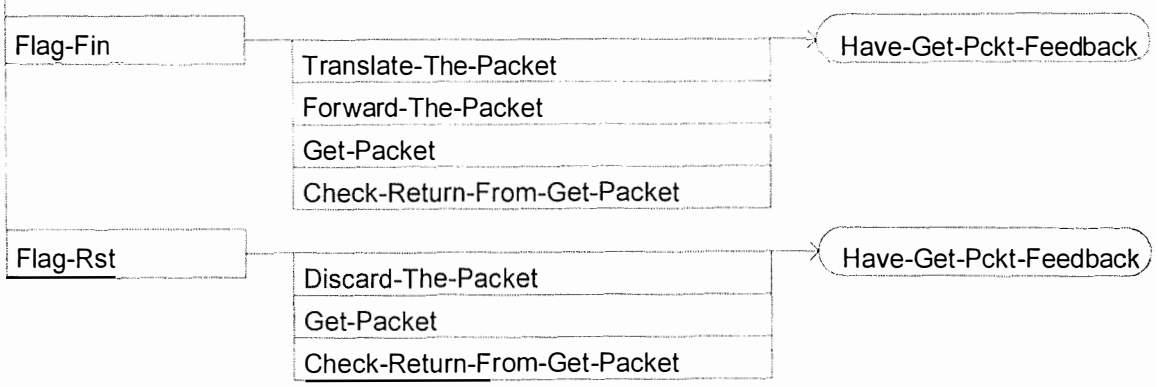



Forward-The-Packet
Get-Packet
Check-Return-From-Get-Packet
Port-Not-Available
Discard-The-Packet
Get-Packet
Check-Return-From-Get-Packet
Have-Get-Pckt-Feedback

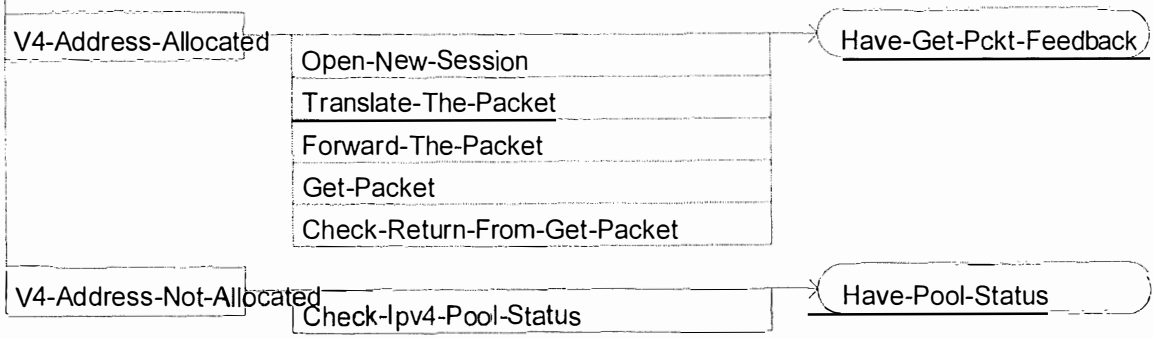
-schema=lrschema.pl



## Have-Flag-Tcp




## Have-V4-Layer-Status




## Have-Pool-Status